Bachelor's Thesis

Grau en Enginyeria en Tecnologies Industrials (GETI)

Python-based Deep-Learning methods for energy consumption forecasting.

Thesis

Author: Josep Roman Cardell

Supervisor: Ramon Costa Castelló

Convocatory: January 2020

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

In a society where we do nothing but increase the use of electricity in our daily life, energy consumption and the corresponding management is a major issue. The prediction of electric energy demand is a key component, for the power system operators, in the management of the electrical grid. The importance of forecasting a particular household daily energy consumption does concern the end-user too, by reason of the design and sizing of a suitable renewable energy system and energy storage.

The aim of this thesis is to develop and train a computing system capable of predicting, with best accuracy as possible, electricity consumption at household-level. This paper presents a Short Term Load Forecasting (STLF) with Artificial Neural Networks (ANN), which lead to accurate results in spite of the dwelling consumption unpredictability. The recorded data, containing the daily track of electricity consumption over a particular household from 2015 to 2018, was analysed. Subsequently, a study over the ANN architecture and training algorithms was carried out in order to define a robust model. Furthermore, several experiments were conducted with different models, containing distinct inputs, aiming to compare the relevance of a diversity of parameters for the network's training. Finally, the forecasting of the optimal models, created with the insights collected over the whole research, was performed and compared in several specially selected time periods.

The results showed how with the appropriate inputs and selection of hyperparameters, a shallow ANN can provide certain accuracy on the forecasting of electric energy demand. As well as a methodology to develop and train an artificial neural network.

Contents

Abstract 1

1 Motivation and Background 9

2 Introduction 11
2.1 State of art . . . 11

3 Artificial Neural Networks (ANN) 13
3.1 Overview . . . 13
3.2 Learning process . . . 14

4 Exploratory Data Analysis (EDA) 19
4.1 Data frame . . . 19
4.2 Target analysis . . . 19
4.3 Outliers . . . 21
4.4 Removing missing data . . . 23
4.5 Correlations . . . 24
4.6 Seasonality . . . 25

5 Building process 35
5.1 The architecture of the network . . . 35
5.2 Data normalization . . . 36
5.3 Initialization . . . 37
5.4 Activation function . . . 37
5.5 Learning algorithm . . . 39
5.6 Number of Epochs and Batch size . . . 40

6 Development Test 43
6.1 Categorical variables . . . 43
6.2 Models . . . 44
6.3 Results . . . 45
6.3.1 1000 iterations experiment . . . 48
6.4 Hyperparameters . . . 50
6.5 Frequency . . . 51

7 Training process 53
7.1 Training/Test split . . . 53
7.2 Modifications . . . 54
7.3 Results . . . 54
7.4 Model 8 . . . 62
7.4.1 Results . . . 68

8 Forecasting 73

9 Cost Analysis 79
9.1 Human labor cost . . . 79

9.2 Equipment costs 79
9.3 Total cost 79

10 Environmental impact **81**

Conclusions and future **83**

Acknowledgments **85**

Bibliography **87**

A Source Code
A I Input and Output
A II Hyperparameter Searching
A III Training and Test

B Additional results
B I Frequency Test
B II Batch size and Epochs

List of Figures

3.1 General Structure of an ANN, source: [Raj19]. 14
3.2 Activation of a neuron, source: [Nav19]. 15

4.1 Target distribution. The blue curve a density probability function, the total area under the curve integrates in to one. The y axis indicate the value of the probability density, while the x axis indicates the real consumption values. The vertical lines; the red one, signals the mean of the distribution while both yellow ones indicate the limits were the 95% of the values are found. 20
4.2 Normal distribution 21
4.3 Box Plot of Monthly Consumption 22
4.4 Box Plot of Weekly Consumption 22
4.5 Daily consumption, with identification of outliers. 23
4.6 Percentage of removed data. 24
4.7 Rolling mean of the total consumption [kWm], window of 7 days. 26
4.8 Monthly Seasonality of all years. 27
4.9 Monthly Seasonality divided by years. 27
4.10 Drop October 2016 28
4.11 Drop May 2016 29
4.12 Drop May 2017 29
4.13 Daily consumption on June divided by years. 30
4.14 July 2016 31
4.15 Stand by periods. 31
4.16 Mean of consumption by day of the week. 32
4.17 Mean of consumption by day of the week divided by years. 32
4.18 Monthly consumption 33
4.19 Mean of consumption per minute group by year. 33

5.1 Sigmoid function, source: [V17] 38
5.2 Tanh function, source: [V17] 38
5.3 ReLU function, source: [V17] 38
5.4 Decreasing of the value of the loss function per epoch. 41

6.1 Distribution of the MAPE for 1000 simulations. The blue curve a density probability function, the total area under the curve integrates in to one. The y axis indicate the value of the probability density, while the x axis indicates the MAPE values. The vertical lines; the red one, signals the mean of the distribution while both yellow ones indicate the limits were the 95% of the values are found. 49
6.2 Box plot of the MAPE for 1000 simulations 49
6.3 Combinations of Hyperparameters. 50

7.1 Learning Rate during the training of model 5. 55
7.2 Scatter plot of the predicted Vs the real values. 56
7.3 Scatter plot of the predicted Vs the real values, Model 7. 57
7.4 Forecasted Vs Real values, 1000 points. 58

7.5 Forecasted Vs Real values, 400 points. 58
7.6 Forecasted Vs Real values, 200 points. 59
7.7 Forecasted Vs Real values, 1000 points. 59
7.8 Forecasted Vs Real values, 400 points. 60
7.9 Forecasted Vs Real values, 200 points. 60
7.10 Forecasted Vs Real values, 1000 points. 61
7.11 Forecasted Vs Real values, 400 points. 61
7.12 Forecasted Vs Real values, 200 points. 62
7.13 Real values Vs the 5 models, 150 points. 64
7.14 Model 8.1 Vs Model 8.3, 1000 points. 64
7.15 Model 8.1 Vs Model 8.3, 1000 points. 65
7.16 Distribution forecasted of values for Model 8.1. The blue curve is the density probability function, the total area under the curve integrates in to one. The y axis indicate the value of the probability density, while the x consumption values. The vertical lines; the red one, signals the mean of the distribution while both yellow ones indicate the limits were the 95% of the values are found. 66
7.17 Distribution of forecasted values for Model 8.2. 66
7.18 Distribution of forecasted values for Model 8.3. 67
7.19 Distribution of forecasted values for Model 8.4. 67
7.20 Distribution of forecasted values for Model 8.5. 67
7.21 Distribution of the real values. 68
7.22 Scatter Plot: Actual values Vs the predicted values for Model 8. 69
7.23 Forecasted Vs Real values, 200 points. 70
7.24 Forecasted Vs Real values, 400 points. 70
7.25 Forecasted Vs Real values, 1000 points. 71

8.1 First week of May 2016, forecasted by Model 7 and 8 and compared to the actual values. 73
8.2 Days 22nd, 23rd and 24th of May 2016, forecasted by Model 7 and 8 and compared to the actual values. 74
8.3 Drop on last week of May 2016, forecasted by Model 7 and 8 and compared to the actual values. 74
8.4 Third week of June 2017, forecasted by Model 7 and 8 and compared to the actual values. 75
8.5 Days 12th, 13th and 14th of June 2017, forecasted by Model 7 and 8 and compared to the actual values. 76
8.6 Peak on second week of June 2017, forecasted by Model 7 and 8 and compared to the actual values. 76
8.7 Last days of the data set followed by the following 24h, forecasted by Model 8. 77

List of Tables

0.1 List of acronyms. 7

4.1 Description of the target 20
4.2 Weekend Correlation. 25
4.3 Festivities Correlation. 25
4.4 Correlation with previous days 25

6.1 Best five MAPEs of Model 7. 46
6.2 Results of first models 47
6.3 Results of last models 48
6.4 Coefficient values. 48
6.5 Statistical description. 48
6.6 Hyperparameters selected for the training. 51

7.1 First target values of the Test set. 54
7.2 Training performance. 55
7.3 Test performance 56
7.4 Training performance of Models 8 63
7.5 Forecasting performance of Model 8 68

9.1 Personal cost. 79

Glossary

Acronym	Definition
STLF	Short Term Load Forecasting
ANN	Artificial Neural Network
DNN	Deep Neural Network
EU	European Union
GPU	Graphic Processing Unit
CNN	Convolution Neural Networks
RNN	Recurrent Neural Networks
IL	Input Layer
OL	Output Layer
HL	Hidden Layer
HN	Hidden Node
w	weights of a connection between two nodes
b	bias of a node
E	Error Function
GD	Gradient Descent
SGD	Stochastic Gradient Descent
EDA	Exploratory Data Analysis
CSV	Comma-Separated Values
std	Standard deviation
LO	Lower Outlier
UO	Upper Outlier
IQR	Interquartile Range
SO	Stochastic Optimizer
MLP	Multy-Layer Perceptron
ReLU	Rectified Linear Unit
lr	Learning Rate
OHE	One-Hot Encoding
MAPE	Mean Absolute Percentage Error

Table 0.1: List of acronyms.

1 Motivation and Background

Nowadays, as we try to improve the way we live, technology is being implemented in all possible aspects of our daily life. One of the most affected fields where technology has inquired on is the production and consumption of electricity. As new appliances are being introduced in our homes, works even at city level, more electricity is required for these devices. Therefore, the accurate forecast of demand at the individual household-level is able to improve the way distribution network operators are dealing with the changes appeared in energy consumption. Another important aspect that is clearly changing in our daily life, is how all sorts of data are being tracked. Consequently, using this exabytes of data combined with deep-learning methods to accurately predict the consumption of energy can lead to significant cost-minimizing and environmentally-friendly changes on electricity management.

Aside from the objective of the study, this thesis represents a personal purpose too. The Machine-Learning and Data-Mining fields, which are exponentially gaining importance in all sorts of aspects of our life, require a diversity of professional profiles to better understanding of the emerging problems as well as their optimal solutions. Therefore, to conclude with the wide range of knowledge in divers fields that the bachelor's degree in Industrial Technologies Engineering has provided me, I wanted to have a first contact with these topics.

The project has its origin on the proposal of the Prof. Ramon Costa Castelló, from the department of Systems Engineering, Automation and Industrial Computing of the polytechnic university of Catalonia, supervisor of this thesis.

2 Introduction

The residential sector presents, in terms of energy consumption, an important part of the total electricity demand. Actually, in 2017 households accounted for 27.2% of final energy consumption in the European Union (EU). Where, most of the EU final energy consumption at households is covered by natural gas (36.0%) and electricity (24.1%) [Eur]. Consequently, apart from the forecasting of the demand at a large scale, which the electrical grid operators are already performing, the prediction has to be done at a smaller scale too.

The aim of this paper is to predict the electric energy demand in a particular dwelling with a period of 24 hours in advance achieving a certain accuracy. This reference was selected as the energy price in the day-ahead market, for several EU countries, is set for each hourly interval with one day in advance. In Spain the auction takes place once a day; at 12pm the auction is conducted for the 24 hours of the next day [Com16].

The forecasting was performed with a stochastic method. Through the following sections, the selection of the specific structure and models for the ANN is explained. Followed by the training process of the network and the corresponding results.

The data provided, corresponds to the consumption of a four members family living in the city of Barcelona. The data was tracked minute by minute, by a sensor with 90% of accuracy, during the years 2015, 2016, 2017 and 2018. As a general overview, the consumption habits at the dwelling were; one washing machine a day, use of an electric coffee machine each morning, use of dishwasher some days a week and use of AC in standard mode some days during the months of July and August.

2.1 State of art

With the improvements in computational power and the use of GPUs[1] came a huge advance in ANN development and applications. From the discovery of the perceptron (1958) and the multi-layer perceptrons (1965), to the introduction of more complex models; convolution neural networks (CNN) and recurrent neural networks (RNN) [Bag18]. The most important advance in this field came with the achievement of deep-learning in neural networks, which led the training of hundreds of layers and so, the exploration and improvement of the existing networks. This advances have had a direct impact in the actual technology sector. Some of this application are; Voice recognition,

[1] Graphic Processing Unit.

image classification, time series forecasting, medical diagnosis...

3 Artificial Neural Networks (ANN)

In this section a brief presentation of ANNs is provided. An Artificial Neural Network is an information processing paradigm inspired on the human brain. It is a biological simulation done on the computer to perform certain specific tasks like clustering, classification and pattern recognition. Inspired by the human brain, ANNs are able to learn from and generalize from experience. One major application area of ANNs is forecasting. By reason of; first, they learn from examples and capture subtle functional relationship among data even if the underlying relationships are unknown or hard to describe. Second, neural networks can generalize, they can infer the unseen part, predictions of future behavior, of the population. Third, ANNs are universal functional approximators, they have more general and flexible forms than the traditional statistical methods. Finally, they are capable of performing nonlinear modeling without a priory knowledge about the relationships between input and output variables [Guo14a].

3.1 Overview

This general overview is focused on a particular structure of ANNs, the multi-layer feedforward networks, which is the most popular and widely-used network paradigm in many applications including forecasting.

ANNs are a supervised learning system built of a large number of neurons, or perceptrons. Each basic unit can make simple decisions and feed those decision to other neurons, organized in interconnected layers. Each of this connections between neurons have an associated weight (w), representing the importance of the corresponding connection to the final result. A shallow network has only three layers of neurons; an input layer (IL), one hidden layer (HL) and an output layer (OL). Meanwhile, a Deep Neural Network(DNN) has two or more HL. Thus, DNNs are more accurate in solving complex problems.

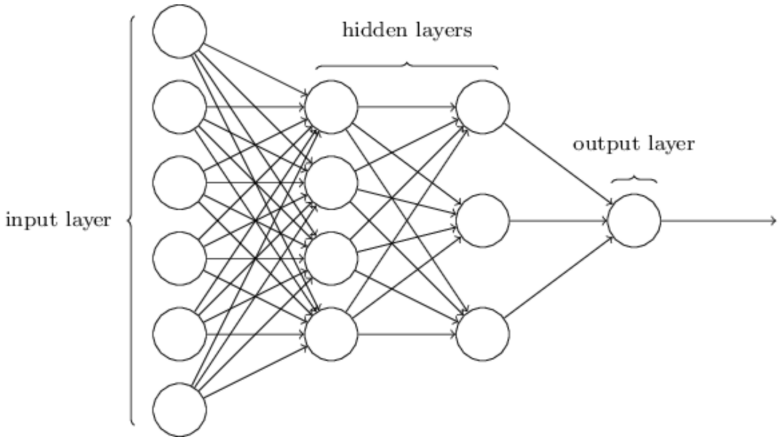


Figure 3.1: General Structure of an ANN, source: [Raj19].

3.2 Learning process

Forward propagation:

First of all, weights are initialized and inputs, x, are fed into the network as the first activation. Then, each neuron computes the sum of the weighted activations of all the neurons in the previous layer, and then add some bias. This linear function, $z(W, b)$, is then followed by an activation function, $a(z)$ which results in the output, that is the activation for next-layer neurons. This process is performed by all neurons in all layers until the output layer, whose neuron output, is the result of the network.

$$z_i^k(W, b) = \sum_{j=1}^{n^{(k-1)}} (w_{i,j}^k * a_j^{(k-1))}) + b_i^k \tag{3.1}$$

$$a_i^k(z) = g(z_i^k) \tag{3.2}$$

Where a_i^k is the output of the neuron i of layer k[2] and $n^{(k-1)}$ is the number of neurons in the previous layer. $w_{i,j}^k$ is the weight of the connection between neuron j in layer $(k-1)$ and neuron i in layer k. $g(x)$ is the activation function. b_i^k is the bias of neuron i on layer k.

Activation function:

An activation function is a mathematical equation which from the weighted sum of input and biases, determines the output of a neuron, it dictates if the neuron can be fired or not. ANNs rely on nonlinear activation functions, as the derivative of the activation function helps learn complex patterns in data through the backpropagation process[3] [CM18].

[2]The inputs of the network, are also known as the activations of layer 0, a_i^0.

[3]Common method of training a neural net in which the initial system's output is compared to the desired output, then the system is adjusted until the difference between the two outputs is minimized.

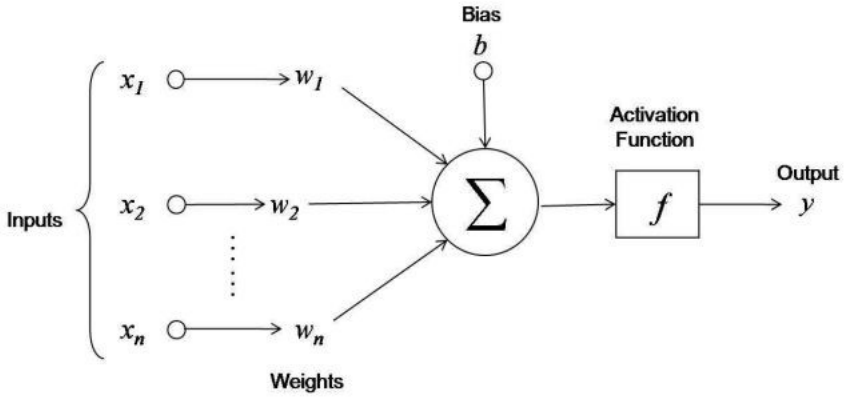


Figure 3.2: Activation of a neuron, source: [Nav19].

Loss function:

When the forward process is performed to generate the initial prediction, there is an error function (E), the *Loss Function*, which defines how far the result is from the actual value. There are two commonly used Loss functions;

$$E(y, \hat{y}) = -(y * log(\hat{y}) + +(1 - y) * log(1 - \hat{y})) \tag{3.3}$$

$$E(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 \tag{3.4}$$

Where m is the total number of training examples. y is the target(actual) and $\hat{y}$ the predicted value. Each equation serves to a particular type of problem. Hence, ***Eq.3.3***, *Cross Entropy*, is used in binary classification problems, while ***Eq.3.4***, *Mean Squared Error*[4], is used for regression tasks.

The goal then, becomes to find a set of weights that minimizes the value of E across the whole training set. The mean of the *Loss function* applied to all samples of the training test is known as *Cost function*. Individual weight's influence on the loss function is determined via backpropagation.

Backwards propagation:

In order to achieve the optimal set of weights, the backward pass is performed, moving back from the network's prediction to the neurons that generated that prediction. Simply stated, backpropagation is a method for calculating the first derivative of the error function with respect to each network weight to find weights that bring the $LossFunction$ to a minimum. It is done by the mathematical process known as $GradientDescent$ (GD)[5]. When the network is learning, it is actually minimizing the $E(w, b)$ by com-

[4]MSE.

[5]It works by iteratively taking small steps towards a lower error value using the gradient of the error function with respect to individual weights.

puting the optimal value for the network parameters (w, b). As the MSE function is a convex functions[6], the aim is to find the global optimum with the GD, by iteratively update the network parameters, (w, b).

The backpropagation algorithm can be decomposed in the following steps :

- Feed-forward computation
- Backpropagation to the output layer
- Backpropagation to the hidden layers
- Parameters updates

The algorithm is stopped when the value of the error function has become sufficiently small [Roj96].

The update of the parameters is done as follows:

$$w_{i,j}^k = w_{i,j}^k - \alpha * \frac{\delta E}{\delta w_{i,j}^k} \tag{3.5}$$

$$b_i^k = b_i^k - \alpha * \frac{\delta E}{\delta b_i^k} \tag{3.6}$$

Where α is a hyperparameter[7] of the network, known as the learning rate, that determines how rapidly the parameters are updated. If the learning is to big, the optimal values will be overshot. In contrast if it is too small, the convergence will require too many iterations.

In order to perform the precedent equations is necessary to apply the chain rule. Consequently, in a simplified way, the steps will be as follow:

$$\frac{\delta E}{\delta z} = \frac{\delta E}{\delta a} * \frac{\delta a}{\delta z} \tag{3.7}$$

$$\frac{\delta E}{\delta w} = \frac{\delta E}{\delta z} * \frac{\delta z}{\delta w} \tag{3.8}$$

Where $\frac{\delta a}{\delta z}$ is the derivative of the activation function.

Finally, from ***Eq.*** 3.1 and 3.8:

$$\frac{\delta E}{\delta w_{i,j}^k} = a_j^{k-1} * \frac{\delta E}{\delta z} \tag{3.9}$$

[6]When the MSE function is passed a value that is unbounded U-shaped (convex) curve is the result. When a bounded value from a Sigmoid function is passed to the MSE function the result is not convex [Kha19]. That is the reason why MSE is commonly used in regression problems but not in classification problems where the values is bounded between 0 and 1.

[7]Known as the parameters defined before the training, which affect the value of the network parameters during the training.

Same process can be done for b.

With the classic GD, the parameters are updated after the whole training set has passed over the forward and backward propagation, but this is usually not practical when working with big sets of data. Typically, a batch of samples is run in one big forward pass, and then backpropagation is performed on the aggregate result. The batch size is an important hyperparameter that is tuned to get the best results. Running the entire training set through the backpropagation process is called an epoch. The Stochastic Gradient Descent (SGD) performs the GD with just one sample per step, in order to avoid redundancy. The process of performing the gradient descent on a batch of samples per step is called the mini-batch Gradient Descent. Mini-batch GD performs an approximation of the gradient from a batch of data points. Therefore, with the advantages from both, GD and SGD, mini-batch can result in stable estimations of the parameters in fewer steps and much faster.

Overfitting and Underfitting

When the neural network is good at learning the training set but do not has the ability to generalize that gained knowledge to additional, unseen examples, is said that the network is overfitting. It is recognizable for suffering low bias[8] and high variance[9]. It can be avoid with retraining the same network with different initial weights values, with a monitoring of the error after each iteration of the training and stopping the process when it starts to over-fit the data or by adding a dropout[10]. On the other hand, underfitting happens when the neural network is not able to accurately predict for the training set, not to mention for the validation set. This is characterized by high bias and low variance. Underfitting can be avoid by adding more training samples [Koe18].

[8] The existence of deviation between the real value and the prediction.

[9] Error, product of high sensitivity to small fluctuations.

[10] Hyperparameter that randomly stops from learning a certain percentage of neurons in every training iteration. This ensures some information learned is randomly removed, reducing the risk of overfitting

4 Exploratory Data Analysis (EDA)

During the EDA the first contact with the data took place. Statistical calculation were performed as well as the creation of plots to find out trends, anomalies, unseen patterns and correlations within the data set. An EDA gives hidden information about the data which is of utmost importance, in this case for the future steps related with the building and training of the ANN.

4.1 Data frame

Initially, the data frame was a CSV[11] file with two types of data, the Timestamp[12] and the consumption values, both containing information from the first day of 2014 to the last day of 2018. The timestamp was given in the following format; "YYYY-MM-DD hh:mm:ss", and represented the precise moment at which the sensor took a measure of the power being consumed. Regarding the consumption, was shown by a number, representing the amount of power being consumed. The data points frequency were minutes and the consumption was registered in Watts per minute (Wm), what is the same, the energy consumed in every minute.

First thing to do was to restructure the data so that it was more tractable. Therefore, the 2.103.888 data points were took and summed up daily, reducing the shape of the array to 1.462. The originated data frame was labeled as *dc* (Daily Consumption.

4.2 Target analysis

The target analysis represents the first contact with the object of study. It brings up some descriptive information about the distribution, variability, outliers... The following table is summary of the basic characteristics of the consumption values;

[11] Coma-Separated Values
[12] Sequence of characters that identifies when an event occurred.

	Consumption(kWm)
count	1462.000000
mean	376.475413
std	117.657293
min	0.000000
25%	320.536000
50%	369.593500
75%	434.889000
max	847.127000

Table 4.1: Description of the target

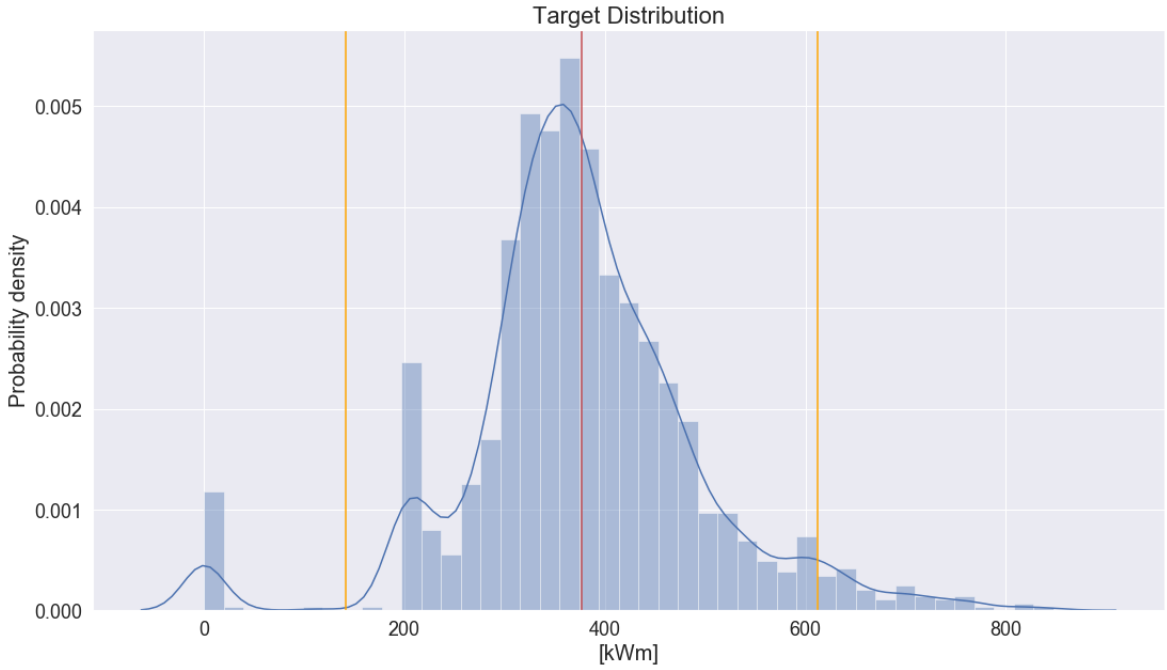


Figure 4.1: Target distribution. The blue curve a density probability function, the total area under the curve integrates in to one. The y axis indicate the value of the probability density, while the x axis indicates the real consumption values. The vertical lines; the red one, signals the mean of the distribution while both yellow ones indicate the limits were the 95% of the values are found.

Fig.4.1 represents the distribution of the data points. Broadly speaking, the set did not look like a normal distribution, as it was asymmetric and the left tail was basically focused in two values. On the right side there was a wider range of outliers values but with low probability to appear. Was also remarkable the fact of finding an important amount of points, as said before, on the 0 and 200kWm values. Those values became subject of study on following sections of the EDA.

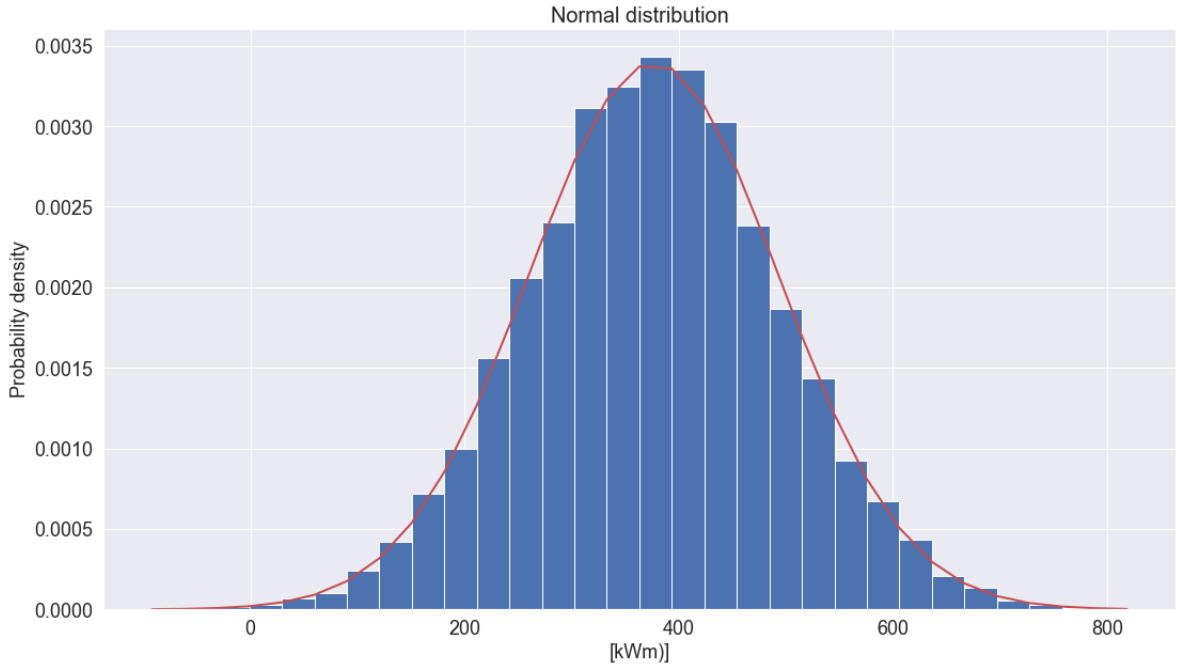


Figure 4.2: Normal distribution

The precedent image is a normal distribution created from a sample of 1000 points, with the actual mean and standard deviation (std) of the consumption values.

4.3 Outliers

The detection of outliers is related with the target analysis, hence information such as the mean or std of the distribution for instance, compiled during the previous section were used in the current section. In statistics, an outlier is a data point that differs significantly from other observations. A data point classified as an outlier can indicate either, high variability in the metered subject or an experimental error, as it can be the disconnection of the sensor.

There are several methods utilized for outliers detection [MSK14], two of them were used during the EDA;

The first procedure is through the visualization of a box plot. It is a method for graphically depicting groups of numerical data through their quartiles. Each "box" is defined by the median[13], 1st and 3rd quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles. The outliers are found further from the median than the whiskers, plotted as individual points.

[13]Middle value of the dataset.

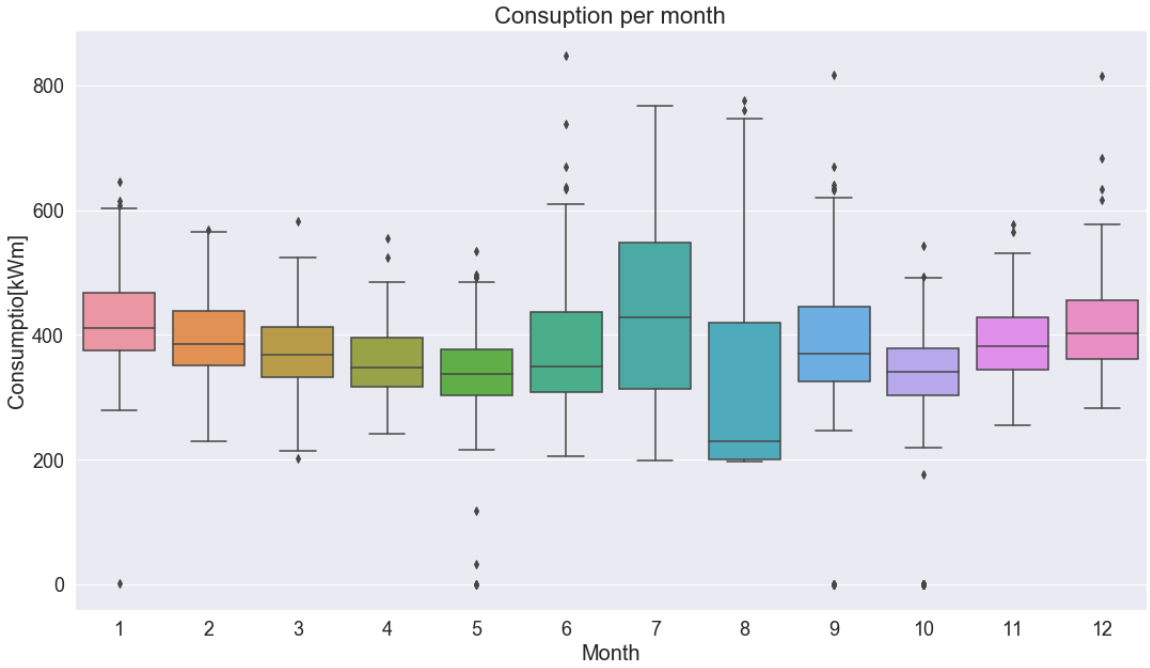


Figure 4.3: Box Plot of Monthly Consumption

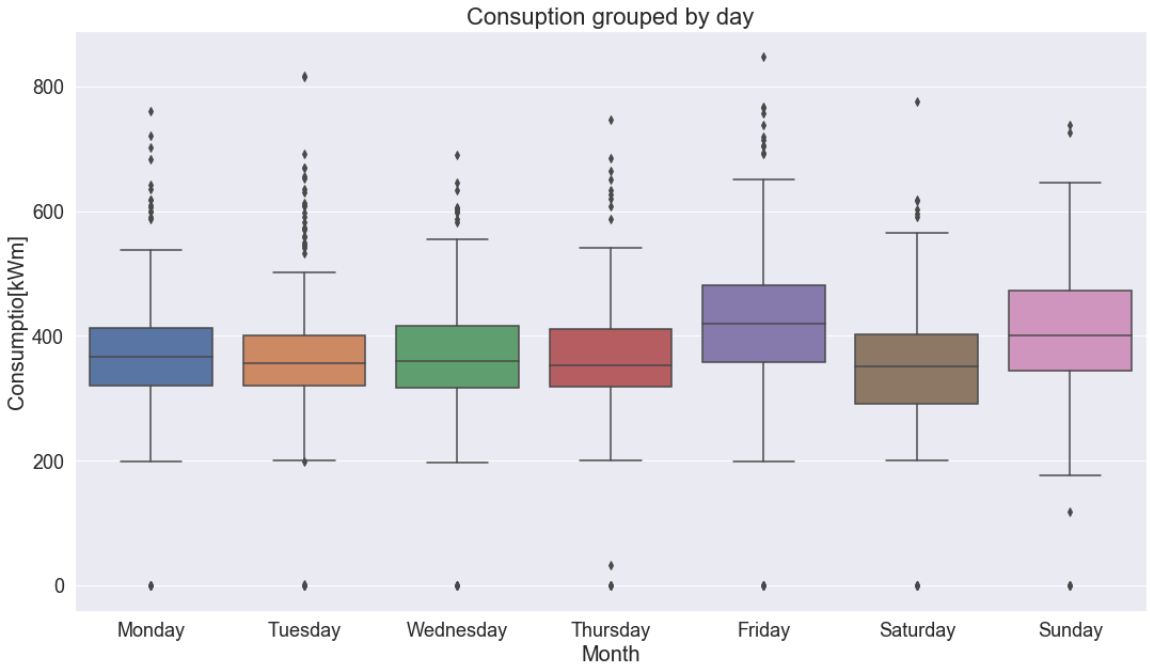


Figure 4.4: Box Plot of Weekly Consumption

Using the Pandas library of python, with the information contained in the dates provided by the timestamp, the data frame was expanded with a label for the day of the week or the month for instance. Both box plots emerged from the mentioned operation.

The second method is using the mathematical formula. The advantage of this procedure is that allows the detection and so elimination if necessary, of the outliers.

So the lower (LO) and upper outliers (UO) are respectively determined by the follow-

ing formulas:

$$LO < (Q1 - 1.5 * IQR) \tag{4.1}$$

$$UO > (Q3 + 1.5 * IQR) \tag{4.2}$$

Where the first and third quartiles were computed during the target analysis. The Interquartile Range (IQR), is a measure of dispersion similar to standard deviation or variance, but much more robust against outliers. And is equal to the difference between the upper and lower quartiles.

$$IQR = Q3 - Q1 \tag{4.3}$$

The threshold values were 149.0065kWm and 606.4185kWm, represented as red horizontal lines in the following plot.

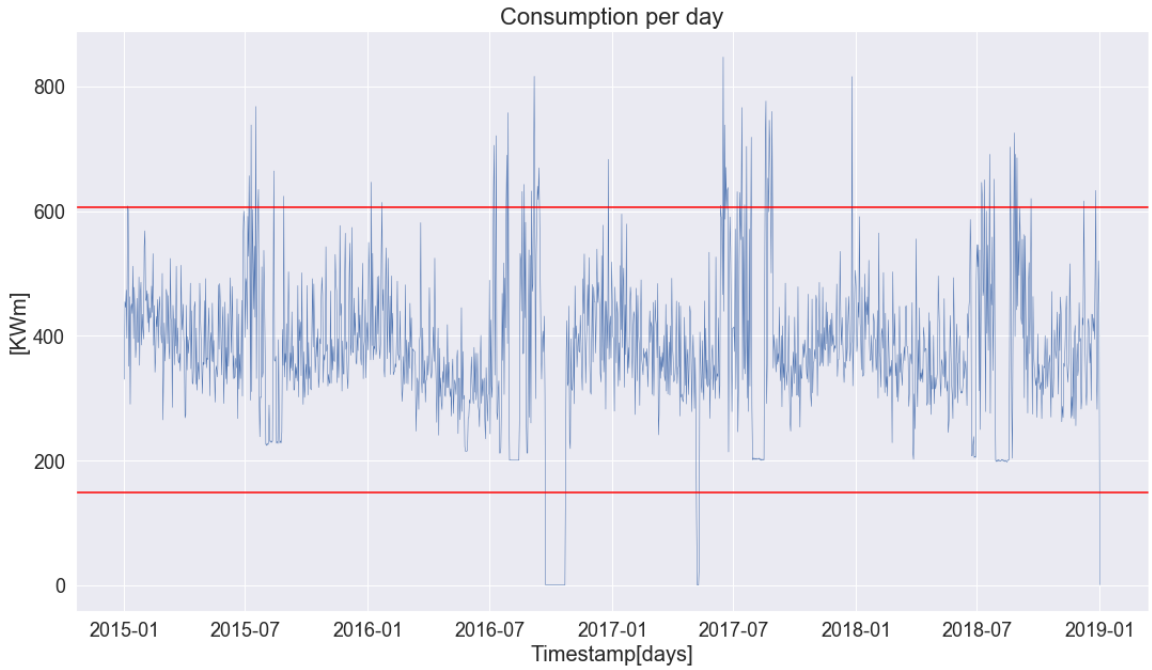


Figure 4.5: Daily consumption, with identification of outliers.

It is important to understand if those extreme values need to be removed or corrected. In this case the values, the outliers found over the upper threshold were not removed for the study as they represent real electric energy consumption values and so the network needs to know of their existence and be trained with them.On the other hand, the outliers found under the lower threshold were removed considering that the only value under 149.0065kWm is 0kWm which is an experimental error.

4.4 Removing missing data

It is always necessary to remove the points which represent an error occurred during the tracking of information, as those are not real values, hence the ANN does not have

to be trained with them. Once it is known of the existence of outliers which need to be removed, it is time to analyse exactly which type of values are they, which percentage they represent and if there is the need of substituting them for other values or not.

Zooming on the drop of October 2016, was seen that, actually, there is not a single value from the 22nd of September to October 23rd of November. This situation led to once reshaping the data set from minutes to hours, days, or months, the Panda's function gave zero value as the sum of non values. So a methodology was performed in order to remove the zeros each time that the data set was reshaped.

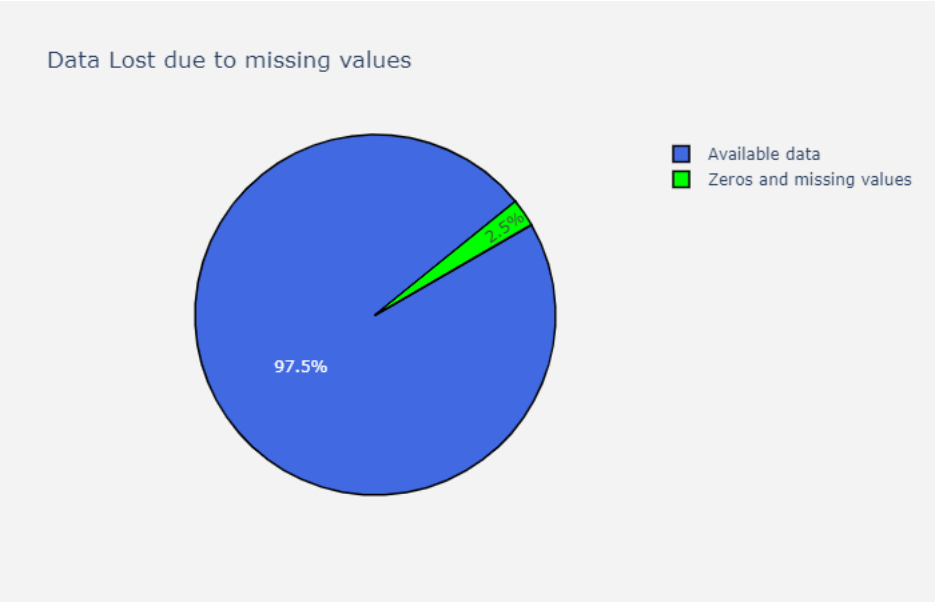


Figure 4.6: Percentage of removed data.

The zeros represented just a 2.5% of the whole set, so there was no need of adding more points on their place, as the remaining data was enough for the experimentation with the network.

4.5 Correlations

The correlation coefficient is not the greatest method to represent the "relevance" of a feature, but it does give an idea of possible relationships within the data. Correlation between features of the dataset and the target can provide insight into which variables may or may not be relevant as input when developing a model to train the network. At first thought, the consumption of a family can be related with either if it is weekday or weekend, if it is a holiday or just a regular day and maybe the season of the year in which the consumption is tracked. All of them are relevant, although, just the first two were discussed in this section as there is another statistical method, by which is easier to analyse seasonality.

In both cases a Point-Biserial Correlation was used to measure the importance of the relations. It is a correlation measure of the strength of association between a continuous-level variable (consumption) and a binary variable (either being weekend or not, or being a festivity day or not) [DeJ19]. Like all Correlation Coefficients (e.g. Pearson's r,

Spearman's rho), the Point-Biserial Correlation Coefficient measures the strength of association of two variables in a single measure ranging from -1 to +1, where -1 indicates a perfect negative association, +1 indicates a perfect positive association and 0 indicates no association at all.

	consumption	weekend
consumption	1.000000	-0.018869
weekend	-0.018869	1.000000

Table 4.2: Weekend Correlation.

	consumption	festivities
consumption	1.000000	0.054197
festivities	0.054197	1.000000

Table 4.3: Festivities Correlation.

Another type of correlation was done in the interest of knowing how the consumption of previous days is connected with the consumption of the target day. So the following values represent the correlation between the value of consumption of previous days with the consumption of the actual day.

	cons.	1_day	2_days	3_days	4_days	5_days	6_days	7_days
cons.	1.0000	0.5249	0.4879	0.4258	0.4030	0.3895	0.3020	0.3961
1_day	0.5249	1.0000	0.5282	0.4926	0.4278	0.4027	0.3902	0.3084
2_days	0.4879	0.5282	1.0000	0.5280	0.4925	0.4283	0.4029	0.3900
3_days	0.4258	0.4926	0.5280	1.0000	0.5281	0.4936	0.4288	0.4020
4_days	0.4030	0.4278	0.4925	0.5281	1.0000	0.5284	0.4936	0.4291
5_days	0.3895	0.4027	0.4283	0.4936	0.5284	1.0000	0.5283	0.4958
6_days	0.302035	0.3902	0.4029	0.4288	0.4936	0.5283	1.0000	0.5297
7_days	0.3961	0.3084	0.3900	0.4020	0.4291	0.4958	0.5297	1.0000

Table 4.4: Correlation with previous days

These results showed a weak correlation between consumption and weekends (***Table*** 4.2) and festivities (***Table*** 4.3). It means that none of the studied cases produce a significant variation, either a rise or descent, on the consumption of electric energy. Regarding the last table (***Table*** 4.4), as it was presumable the relation of the actual consumption with the one at same minute, but on previous days, is decreasing as days get farther.

4.6 Seasonality

Seasonality refers to periodic fluctuations. In time series data[14], seasonality is the presence of variations that occur at specific intervals shorter than a year, such as weekly,

[14]Points of data sequenced in successive order in time.

monthly, or quarterly. It may be caused by various factors, such as weather, vacation, or holidays and consists of periodic, repetitive, and generally regular and predictable patterns at the levels of a time series. There are four common seasonality types: yearly, monthly, weekly, and daily.

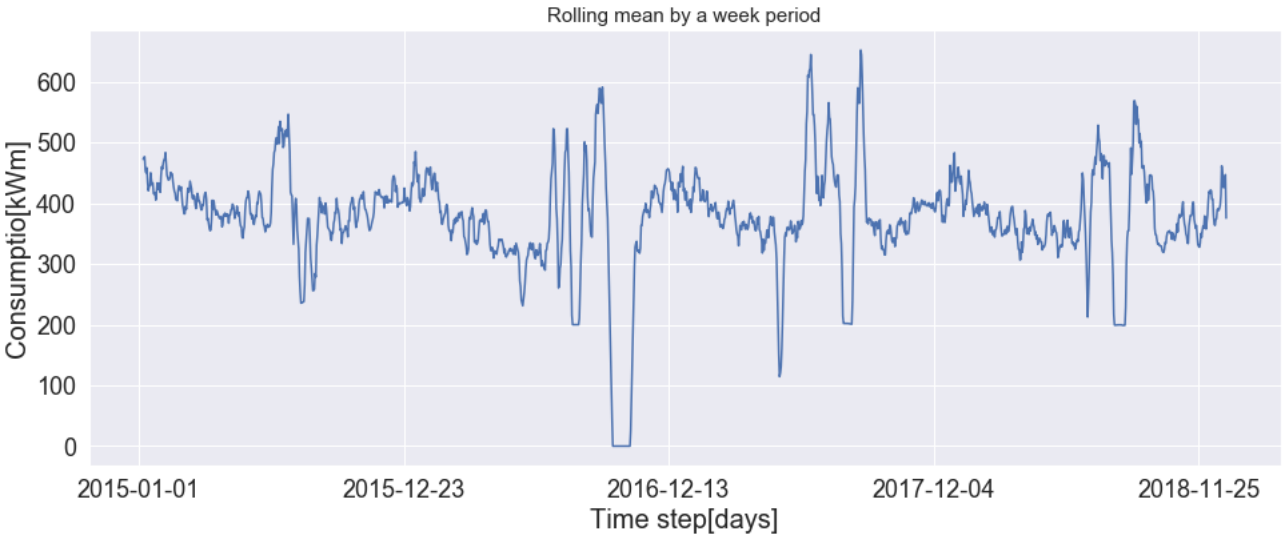


Figure 4.7: Rolling mean of the total consumption [kWm], window of 7 days.

The rolling mean is a way to visualize seasonality, removing part of the outliers and other noise.

Monthly Seasonality:

Firstly, a new data frame[15], was created from the *dc* data frame. It was reshaped monthly, computing the daily mean of each month. So the monthly seasonality is represented on the following plots;

[15]The new data frame was labeled as mm (Monthly Mean)

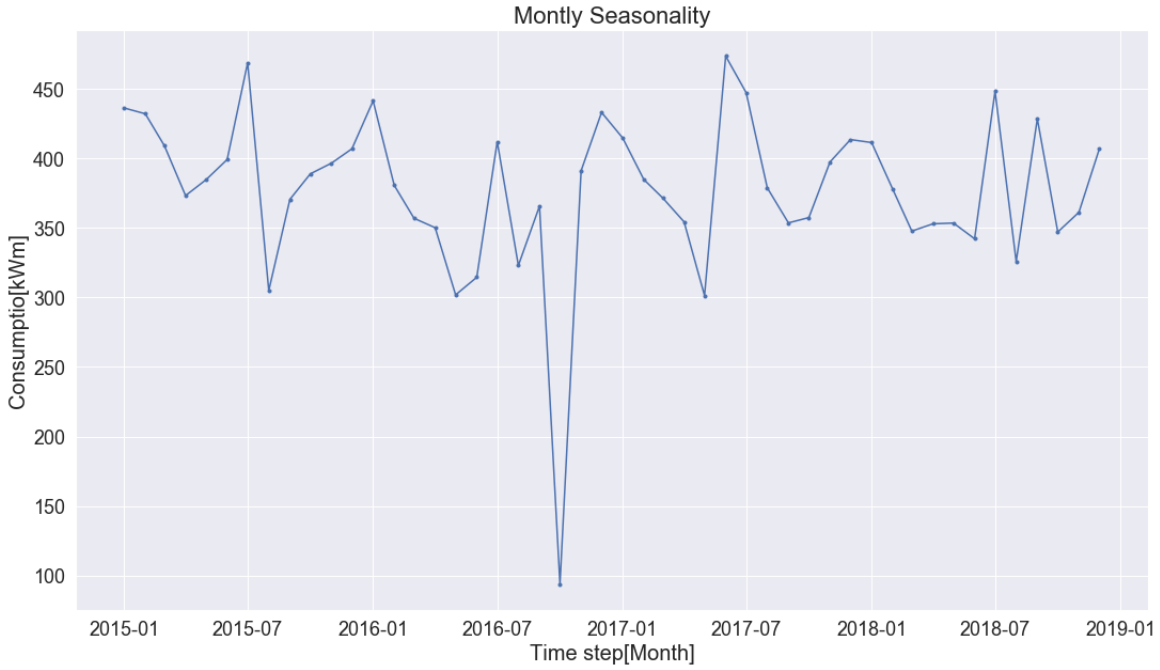
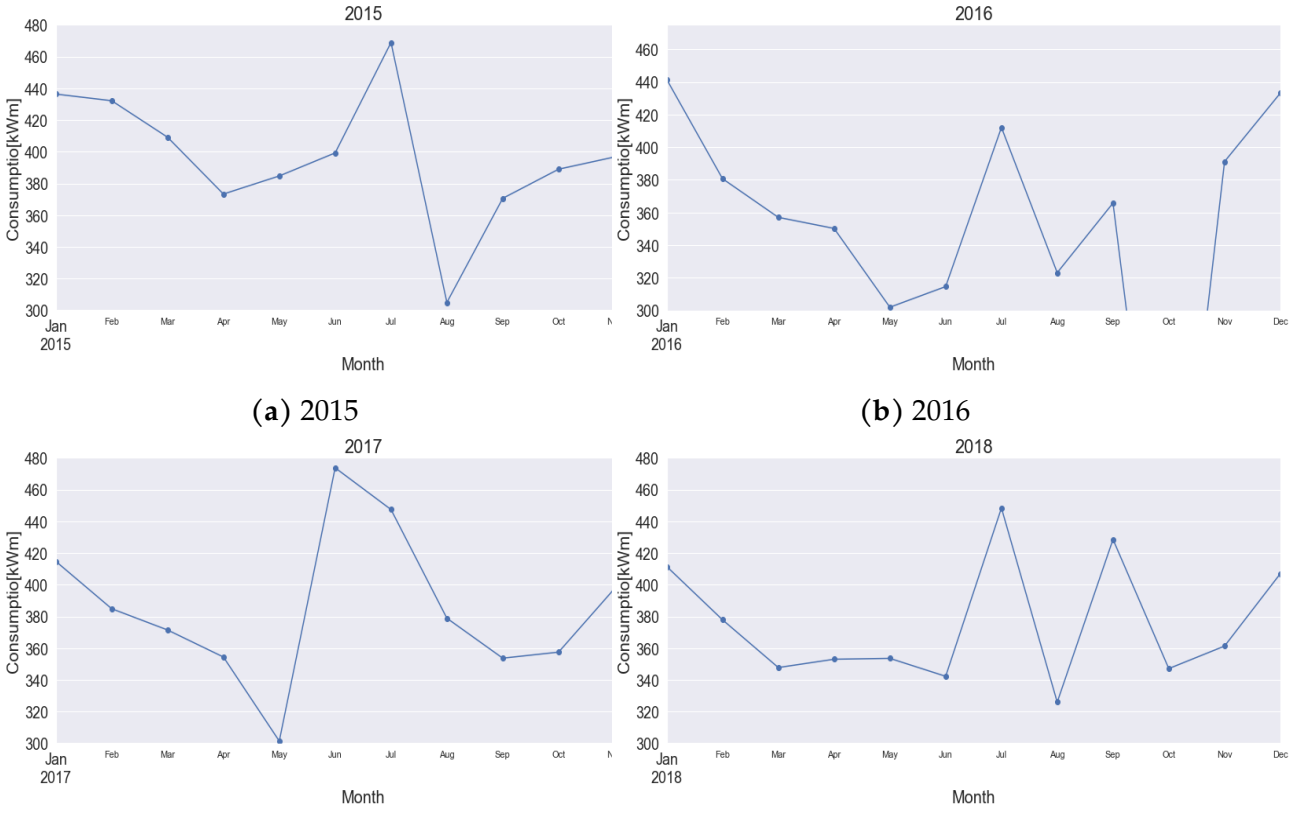


Figure 4.8: Monthly Seasonality of all years.



(**a**) 2015 (**b**) 2016

(**c**) 2016 (**d**) 2016

Figure 4.9: Monthly Seasonality divided by years.

The monthly seasonality from ***Fig.*** 4.10, showed some repetitive patterns over the years as well as some anomalies to zoom on. Focusing (***Fig.***4.9), it can be seen a drop of

the consumption over first trimester of the year, followed by a raise during June and July and another one during the last trimester of the year. At the 2016's plot there is a missing part, October 2016, that is missing because in order to make a first comparison between years, the four years were plotted over the same range of values, from 300kWm to 480kWm. However, the extreme decrease in consumption suffered during October 2016 is clearly represented in the general plot.

Zooming on each year's graphic, it was noticeable that each month, over the four year, had similar values. Despite the similarities, some differences were visible.

In order to zoom in some interesting aspects of consumption trends, the *dc* data frame was more appropriated.

Drop in October 2016:

Probably the most remarkable aspect from ***Fig.***4.7 is the Drop in consumption form October of 2016.

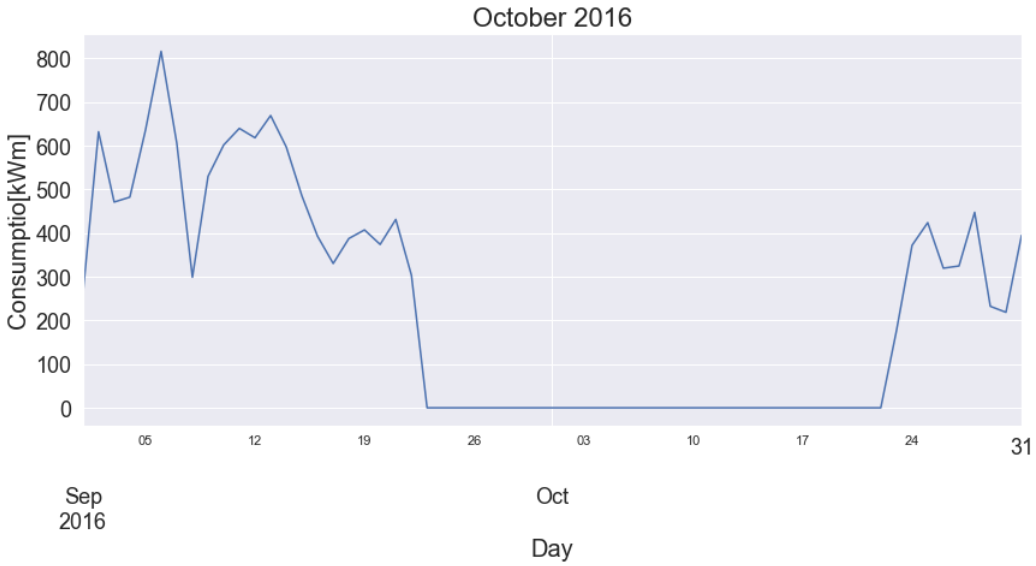


Figure 4.10: Drop October 2016

As commented on the previous section 4.4, from September 23rd to October 22nd the consumption was a constant 0Wm value. Even when anyone is home, the consumption of a house is never absolute zero, as there is always a remaining consumption from plugged devices such as the fridge, freezer, TV,etc. Either if it is to maintain a certain temperature or just for remaining in stand by. So this 0 value recorded by the sensor was caused by a total disconnection from the mains or at least a disconnection of the sensor.

Drop in May 2016:

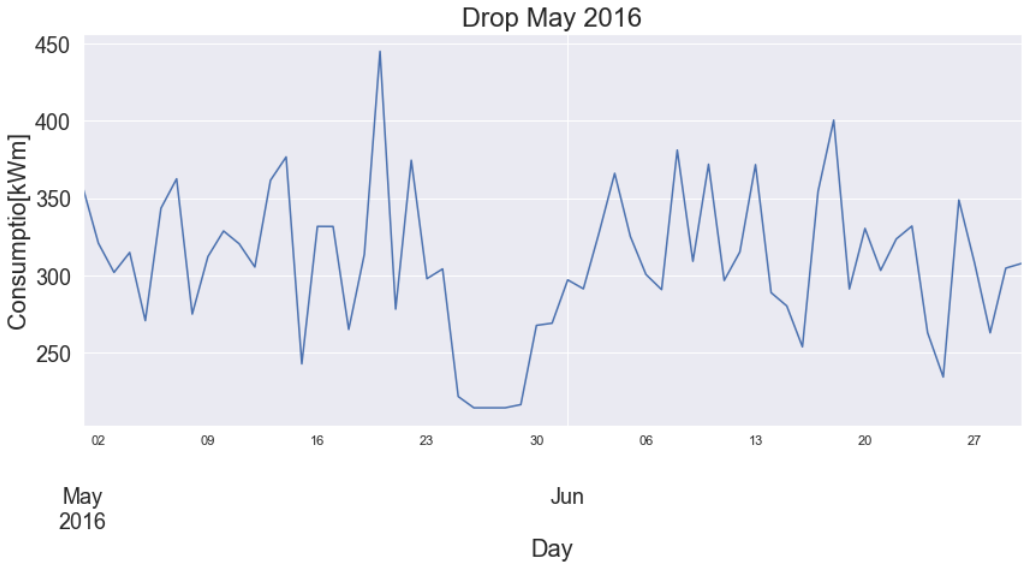


Figure 4.11: Drop May 2016

There was little, but not null, consumption during last week. In general there was lower consumption on May and June of 2016 compared to the other years.

Drop in May 2017:

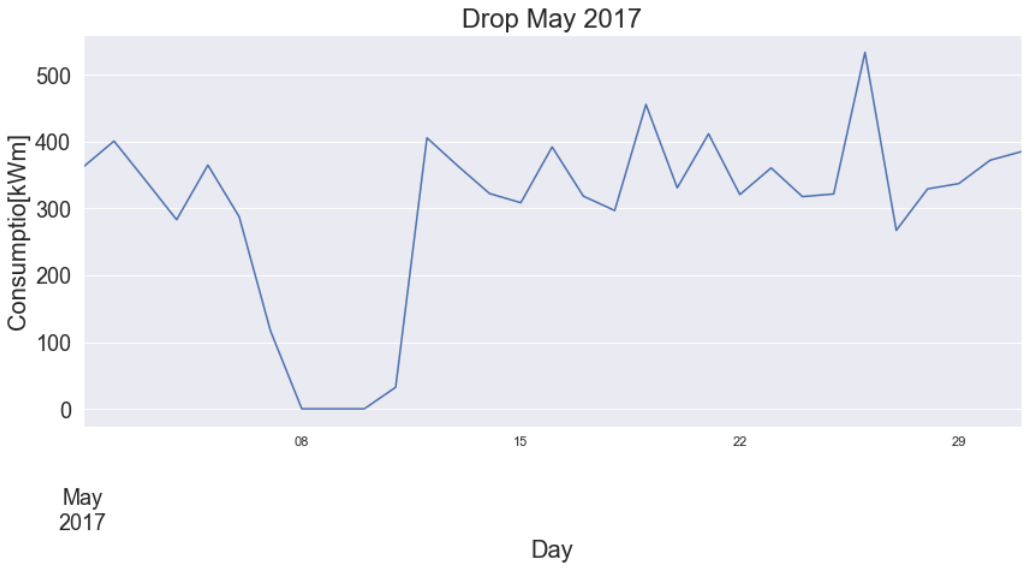
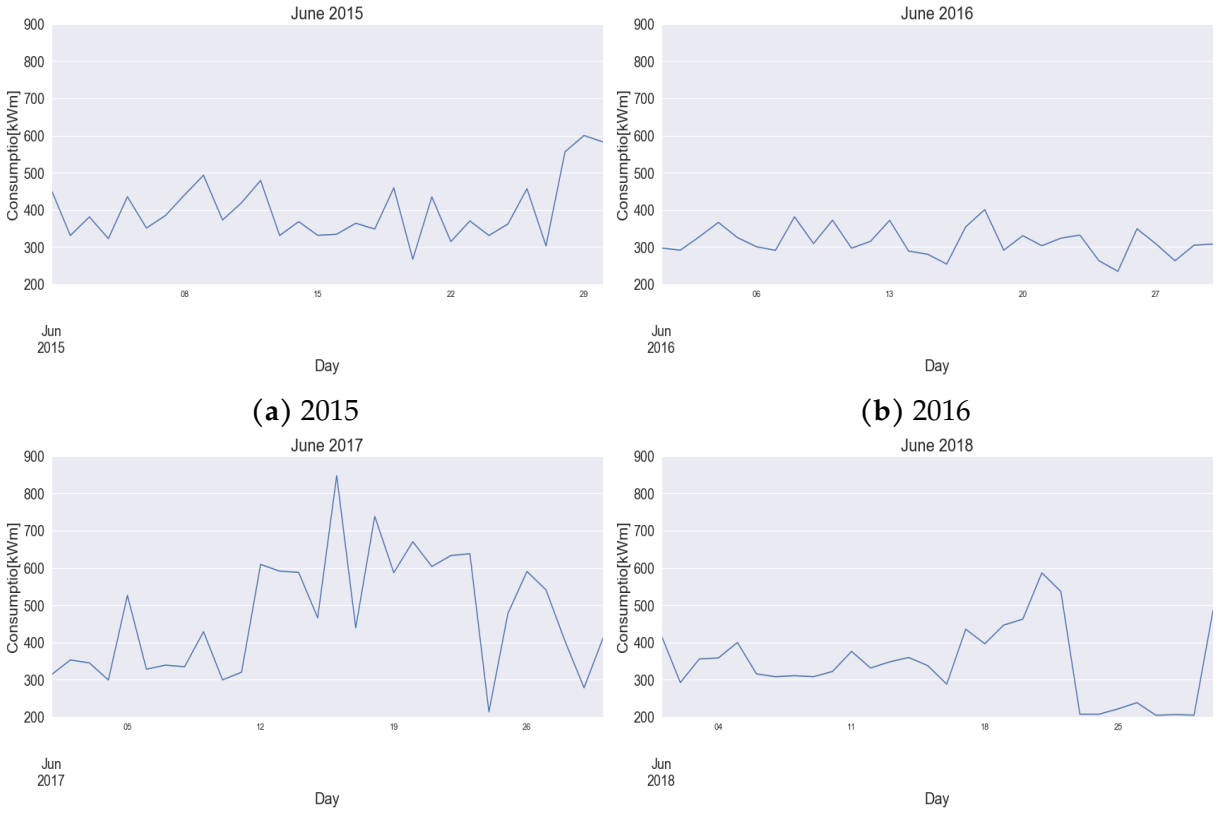


Figure 4.12: Drop May 2017

Same case as in October of 2016 from 8th to 11th of May 2017.

June:

June was plotted for the four years as an unusual month, with different consumption trends each year.



(**a**) 2015 (**b**) 2016

(**c**) 2017 (**d**) 2018

Figure 4.13: Daily consumption on June divided by years.

In 2015 and 2016 the consumption was quite uniform, low variability, with lower values on 2016. In 2017 there was a huge pick in consumption, more variability than in previous years and in general higher consumption. Finally in 2018, the two first weeks where uniform following the 2015 and 2016's trend. Although there was a raise on the third week of the month, on the last week the consumption decrease again till values under the mean of the two first week.

July 2016:

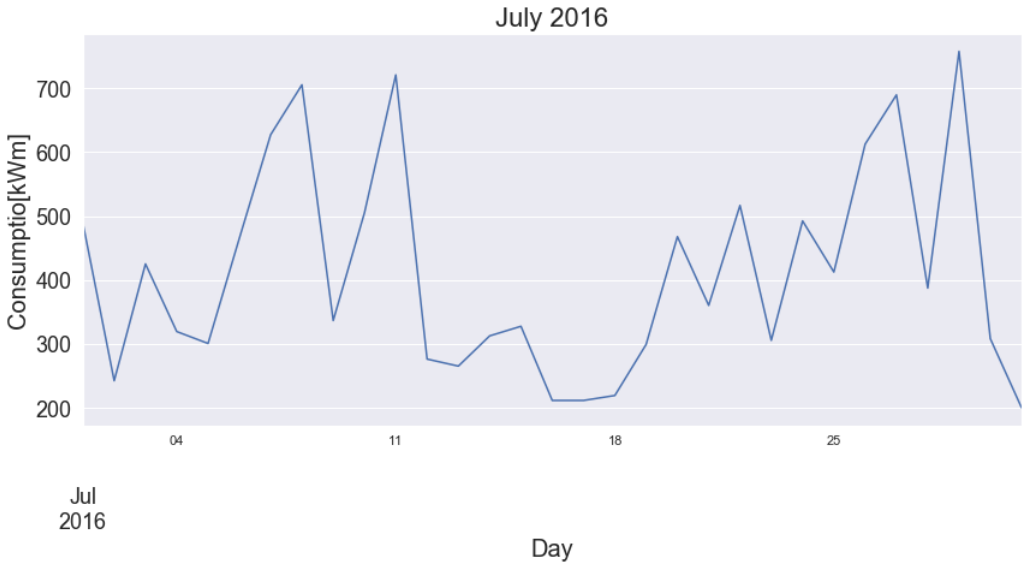
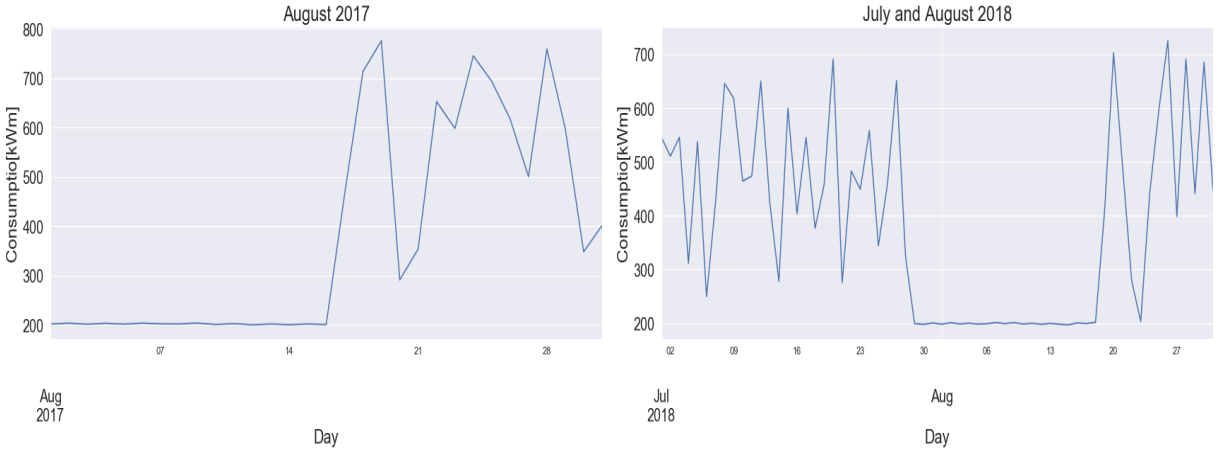


Figure 4.14: July 2016

Lowest consumption of all July was on 2016, may be due to not such a hot summer as June of same year showed also lower values compared to the other years.

The stand by periods:



(**a**) 2015 (**b**) 2016

Figure 4.15: Stand by periods.

Both plots show periods of time with a constant 200 kWm of consumption, assumable to the remaining consumption commented before.

To sum up, there is a perceptible, general, seasonality over the years, which it is worth to take into account when modeling the neural network.

Weekly Seasonality:

An analysis of the week period was performed too. The points from the *dc* data frame were grouped by day of the week and then the mean was computed to each day.

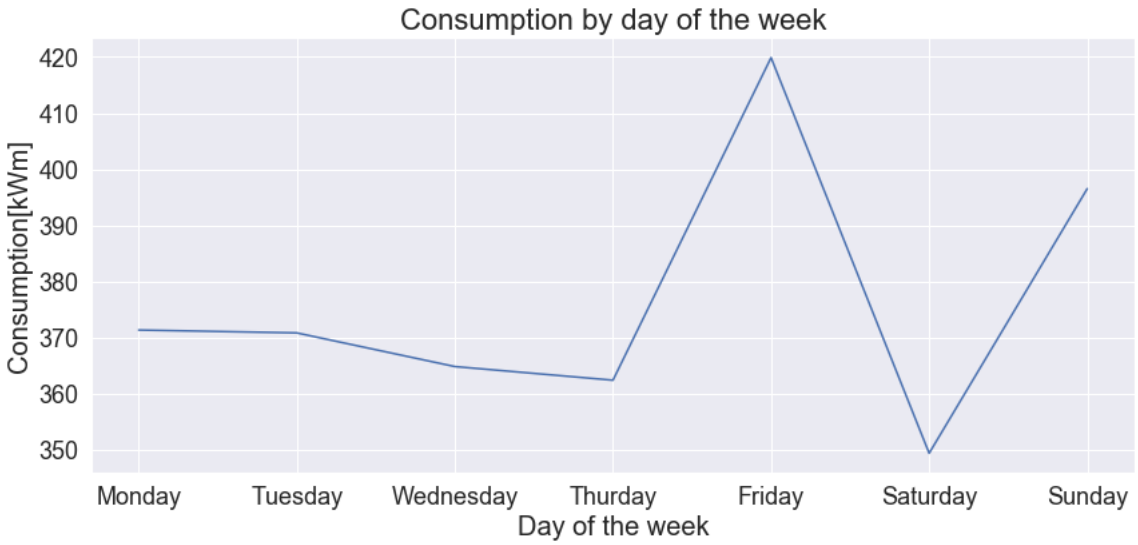
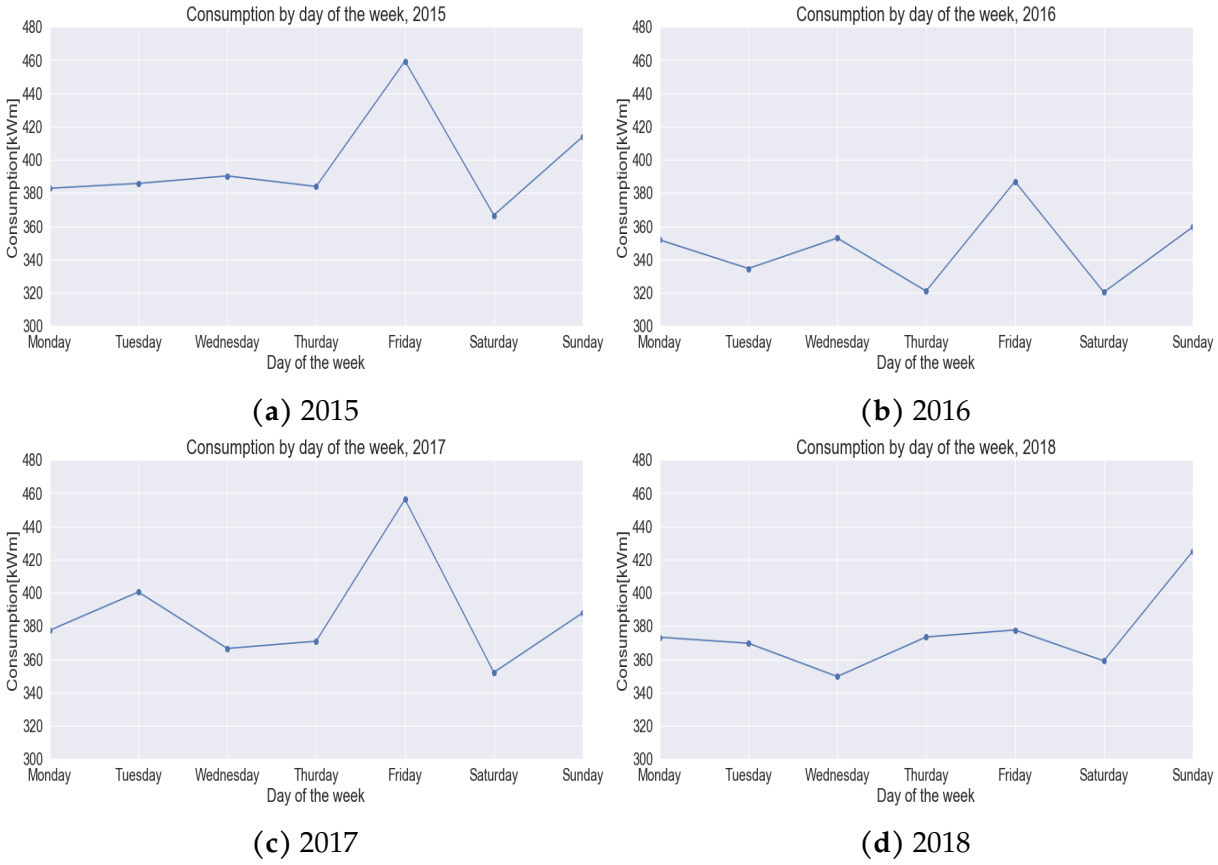


Figure 4.16: Mean of consumption by day of the week.



(**a**) 2015 (**b**) 2016

(**c**) 2017 (**d**) 2018

Figure 4.17: Mean of consumption by day of the week divided by years.

A similar pattern was observed on the 3 first years with a change on 2018 (***Fig.*** 4.17d). The first pattern describes an almost constant value, with small variability, during the first 4 days of the week, with a pick on Friday, descent on Saturday and a rise back on Sunday. On the other hand, in 2018 there was a pretty constant value of consumption with low variability for the whole week with a peak on Sunday.

Comparing years seasonality:

The following plots show, in general perspective, how the consumption of energy has changed during the years for this dwelling.

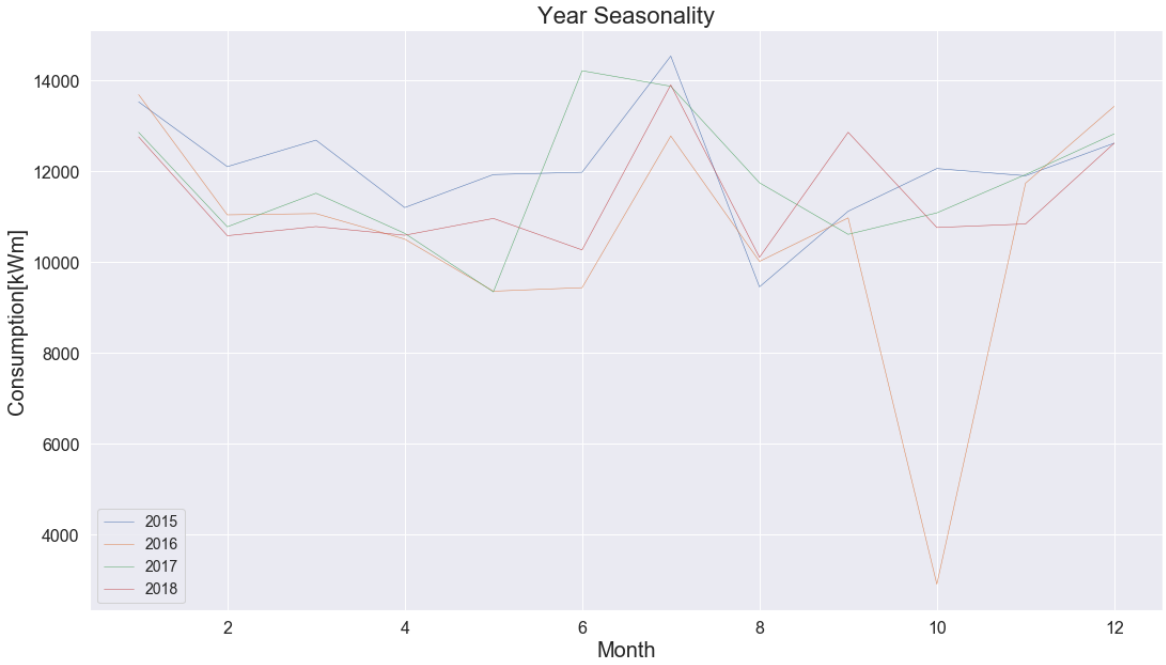


Figure 4.18: Monthly consumption

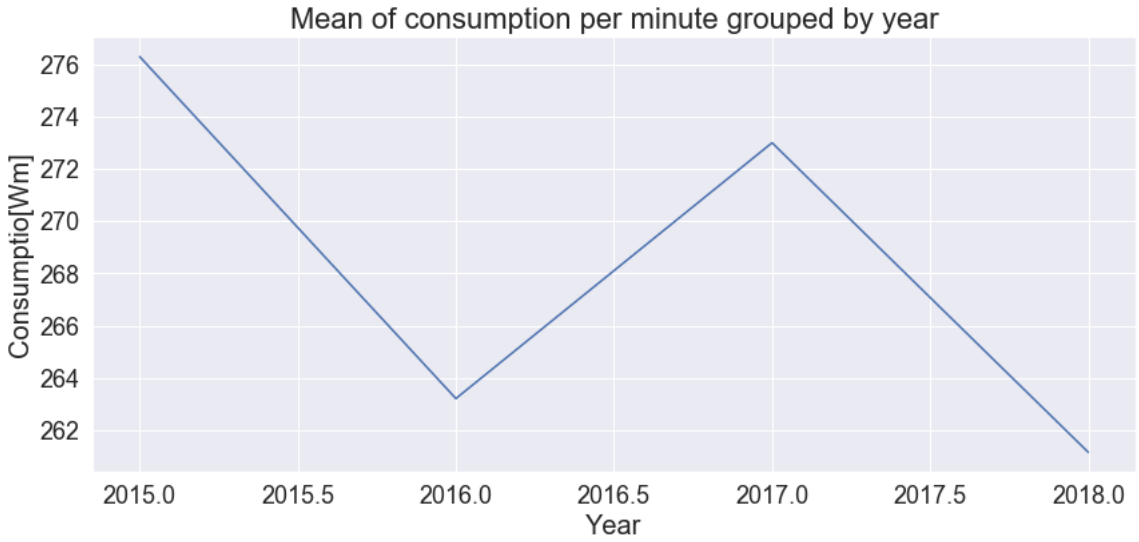


Figure 4.19: Mean of consumption per minute group by year.

On the first image (***Fig.***4.18) the monthly consumption for each of the years is represented, so the difference between each year is perceptible. Apart from punctual aspects, such as the summer raise in consumption of 2017 which started a month before, the years follow a similar trend with a displacement towards lower values of consumption from 2015 to 2018. The second one (***Fig.***4.19), instead, shows the mean of the values of consumption per minute that the sensor has been tracking from 2015 to 2018, both

included. This could be an indicator of how energy technology have improved over years.

5 Building process

Building a neural network for a particular forecasting problem is a nontrivial task. This process can be specified by three entities; interconnections, the activation functions and the learning aspects.

Interconnection can be defined as the way processing elements (neuron) in ANN are connected to each other. These arrangement have two layers which are always common to all network architectures, input layer and output layer. The IL accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer. On the contrary, the OL bring up the information learned by the network. The third type of layer are the HL, which perform all sort of computations on the features entered through the input layer and transfer the result to the output layer. The addition of neurons to the HL, is translated to an boost of the system's computational and processing power, however, the training phenomena of the system gets more complex at the same time.

The purpose of the activation function is to introduce non-linearity into the output of a neuron so that is not just a simple linear-regression problem. The function is activated when the computed result reaches the specified threshold [Mal17]. The input on this instance is the weighted sum plus bias, associated to a particular neuron:

$$Output = activation(x_1w_1 + x_2w_2 + x_3w_3 + ... + bias) \tag{5.1}$$

The way an ANN learns, as explained earlier 3.2, entails several choices which influence the speed and complexity of the training as well as the certanty on the computation of the output values.

The previous concepts are defined by what is known as hyperparameters; external parameters from the model, specified to adjust the network optimizing the accuracy, execution time and memory required. The aim of this section is to gather the necessary information on the modeling of the network, then select a set of the optimal values. Finally perform the development test so that the best combination of the chosen hyperparameters is finally found.

5.1 The architecture of the network

Artificial neural networks are typically composed of layers of nodes, where all the input nodes are in one input layer, all the output nodes are in one output layer and the hidden

nodes are distributed into one or more hidden layers in between. Hence, the design of the structure is defined by: the number of layers and the number of nodes the hidden layer has.

The most reliable way to configure these parameters for each specific forecasting problem is via systematic experimentation [Bro17]. However, this method can be to complicated if lack on experience, as there are so many possibilities.

A single-layer neural network can only be used to represent linearly separable functions, that is way a "Multi-layer Perceptron" (a.k.a MLP) was chosen. With one hidden layer, an MLP can approximate any function that is required. In addition a single HL provides a shallow instead of a deep neural network, which reduces the complexity of the network and so the amount of computation required.

There is a lot of discussion regarding the number of nodes required for each of the hidden layers, and there is not an unarguable valid number. Despite of the controversy, there are some papers related to this topic which claim that fewer hidden nodes are better as they have better generalization[16] and less overfitting. For the observed data, there are an infinite number of functions that pass through all input-output pairs. The "best" function is not necessarily one which fits all of the observed data, but instead is one that generalizes well. In modeling of ANN literature, some well known values for the number of hidden units is either, n, n+1, 2n and n/2 [Guo14b].

Nevertheless, is important to keep in mind that in Deep-Learning each problem is different and so a good starting point is to analyse related problems for testing some ideas.

Taking all this into account during the development test, for almost all the models, one hidden layer was build, one attempt with 2 hidden layers was done too. The choice of the number of hidden units is more complex so all the models have been build with n, 2n and 32 or 50 hidden nodes. The 32 and 50 gave the perspective of the effect of having a much larger value than the n and 2n.

Regarding the other layers, at the beginning of the project was settled that one output was going to be computed each time. The number of nodes at input level depends on the inputs used to feed the network, which was different for each model.

5.2 Data normalization

Data normalization is often performed before the training process begins and it is an important aspect for the further building of the network so it needs to be mentioned in this section too. Why is it so important? Normalization makes training less sensitive to the scale of features. The use of a normalization method will improve analysis from multiple models. In addition, normalizing will ensure that a convergence problem does not have a massive variance, making optimization feasible. There is some debate stating it is better to have the input values centred around 0[17] rather than between 0 and 1 [DeF19], but this is not a topic to deepen now. Focusing on the normalization, all data points were rescaled such that any specific z will now be $0 \leq z \leq 1$, and was done

[16] Generalization is the ability to make successful predictions on unobserved inputs from observed data.

[17] Resulting of the process known as standardization.

through the following formula;

$$z = \frac{x - min(x)}{max(x) - min(x)} \tag{5.2}$$

It is worth to mention that, as a result of normalizing, the observed output of the network correspond to the normalized range. Thus, to interpret the results obtained, those values must be rescaled back to the original range.

5.3 Initialization

As mentioned before, artificial neural networks are trained using a stochastic optimization (SO) algorithm. SO[18] algorithms use randomness in order to find a good enough set of weights for the specific mapping function from inputs to outputs in the data that is being learned. Specifically, SGD requires that the weights of the network are initialized to small random values. Although in general practice biases are initialized with 0, in the hypothetical situation of the weights being initialized to 0 too, all neurons of the same layer would perform the same computation. Therefore, the derivative with respect to loss function would be the same for all the w_i^k, thus all weights would have the same value in subsequent iterations. This make hidden units symmetric and continues for all the n iterations.

There are several initialization methods. One of the most generally used is the Xavier[19] initialization which initialize the weights by drawing them from a Gaussian distribution with zero mean and a specific variance which depends on the number of neurons at the input layer, n, [Xav10].

$$Var(w) = \frac{1}{n} \tag{5.3}$$

Is important to not initialize the weights to large or too small numbers. Otherwise, the gradient of the activation function will be performed on the extremes of the activation function, were the slope is softer, thus the optimization algorithm would slow down.

5.4 Activation function

The activation function is also called transfer function. It determines the relationship between inputs and outputs of a node and a network. The most used activation functions in practice include:

- Linear function
- Sigmoid function

$$g(z) = (1 + e^{-z})^{-1} \tag{5.4}$$

[18] Stochastic optimization methods are optimization methods that generate and use random variables.
[19] Also known as Glorot.

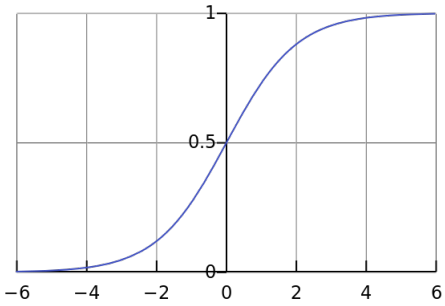


Figure 5.1: Sigmoid function, source: [V17]

- Hyperbolic tangent (tanh) function

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{5.5}$$

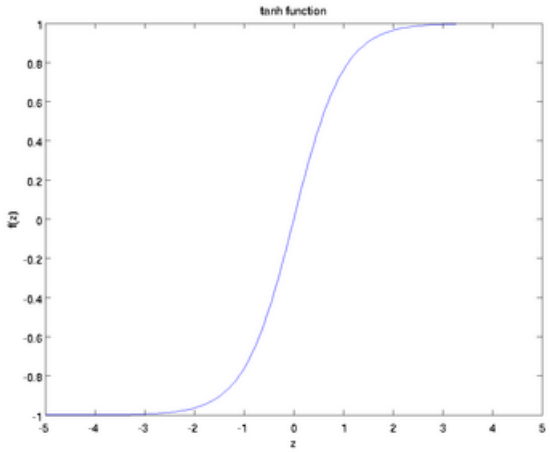


Figure 5.2: Tanh function, source: [V17]

- Rectified Linear Unit (ReLU)

$$g(z) = max(0, z) \tag{5.6}$$

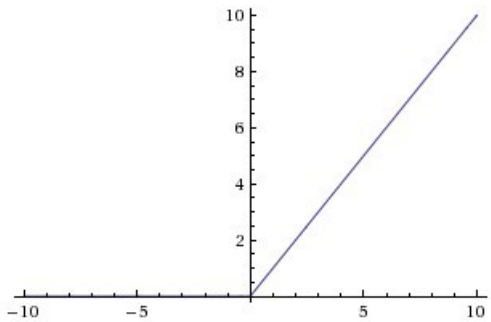


Figure 5.3: ReLU function, source: [V17]

- Sine or cosine function

Generally, a network may have different activation functions for different nodes in the same layer. However, in this case, the same function was used for the entire layer, with a different transfer function for the hidden layer and the output one. Conventionally, the sigmoid activation function seems well suited for the output nodes for many classification problems where the target values are often binary, as the range of values, showed in **Fig.**5.1, is [0,1]. For the problem of study, values have been normalized, ***Section*** 5.2, so even though is a forecasting problem, the logistic function[20] was chosen for the output layer. For the hidden layer, the ReLU, being less computationally expensive than tanh and sigmoid as it involves simpler mathematical operations, and tanh were compared during the development test.

5.5 Learning algorithm

The neural network training is a non linear minimization problem in which the weights of the network are iteratively modified to minimize, in this case, the total squared error between the desired and actual output values for all output nodes. To continue with the same philosophy, there are several optimization methods and non of them guarantee an optimal solution for a general problem. So again, the most popularly used training methods were chosen.

Therefor, the SGD algorithm was selected, ***Section*** 3.2. For this algorithm, a learning rate must be specified which determines the magnitude of the change suffered by the weight on each iteration. The training can be quite sensitive to this hyperparameter as a smaller learning rates tend to slow the learning process while larger ones may cause network oscillations in the weight space. A possible way to improve the learning process is to include two additional terms; momentum, which makes the next weight change in more or less the same direction as the previous one and so to reduce the oscillation effect of larger learning rates, a typical choice of momentum is between 0.5 to 0.9. The other term is the decay, which makes the learning rate, or what is the same the learning step size, to decrease through every iteration, producing a big step size at the beginning of the descent and smaller, more precise, step size at the end while approximating to the optimal value.

Again the best values for the learning parameters are usually determined through experimentation, in between a selected group of values. Among those values, 0.05, 0.01 and 0.001 were compared for the learning rate, while momentum and decay were fixed at 0.9 and 1e-6 respectively.

The challenge of using learning rate schedules[21] is that their hyperparameters have to be defined in advance and they depend heavily on the type of model and problem. By contrast, adaptive gradient descent algorithms such as Adagrad, Adadelta, RMSprop and Adam, provide an alternative to the classical SGD. These learning rate algorithms provide heuristic approach without requiring expensive work in tuning hyperparameters for the learning rate schedule manually [Lau17].

In order to have both methodologies, apart form the SGD in representation of the learn-

[20] The sigmoid function is also known as logistic function.
[21] Methodology by which the learning rate changes during the training.

ing rate schedule, two adaptive gradient descent algorithms were compared during the modeling too;

- Adagrad; is an optimizer adapted to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the learning rate.
- Adam [KB14]

To conclude with the learning aspects the Loss function was chosen; the ones used for default are the "Cross-Entropy", used in classification problems, and the "Mean Squared Error", for regression ones. So as this case requires, the MSE has been used. As a good metric is required to be able to numerically compare different models, the "Mean Absolute Percentage Error" was chosen.

5.6 Number of Epochs and Batch size

The SGD algorithm is iterative. This means that the search process occurs over multiple discrete steps, each step hopefully slightly improving the model parameters. For better understanding lets start with some clarifications; a training dataset is comprised of many rows of data, e.g. many samples. The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. On the other hand, the number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. An epoch is comprised of one or more batches [Bro18].

How this hyperparameters are combined together, is thinkable as a for-loop over the number of epochs where each loop proceeds over the whole training dataset. Within this for-loop there is another one that iterates over each batch of samples, where one batch has the specified “batch size” number of samples. At the end of each batch the predictions are compared to the expected output variables and an error is calculated. From this error, the learning algorithm is used to improve the model, in this case descending over the gradient.

When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent. In the case of mini-batch gradient descent, popular batch sizes include 16, 32 and 64 samples. The number of epochs is traditionally large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently minimized. In the literature the number of epochs is commonly set to 10, 100, 500, 1000, and larger... For the case of study, using a line-plot showing the metric value and the value of the loss function, ***Fig.*** 5.4, over the number of epochs, the following values were selected to be analysed during the development test; 10, 50, 100 and 150.

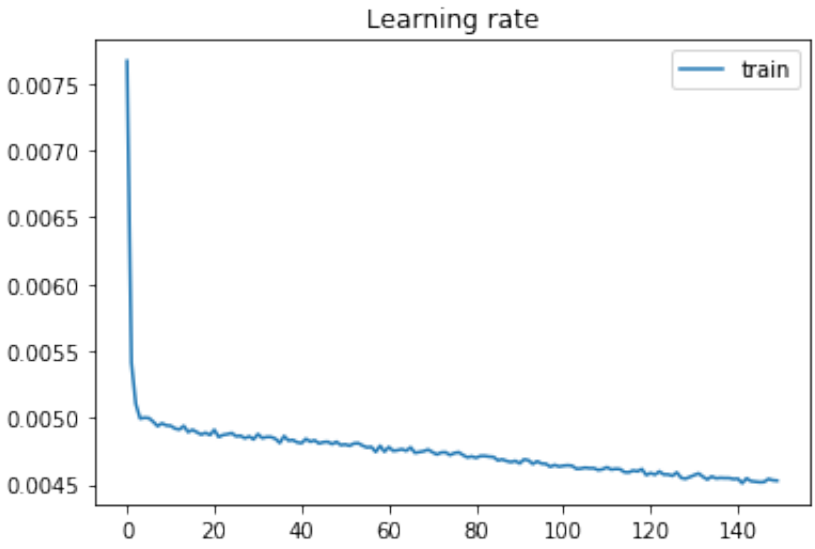


Figure 5.4: Decreasing of the value of the loss function per epoch.

6 Development Test

The Development Test is a process where a representative sample of the set is selected and fed to the network in order to perform the hyperparameter searching, as well as the selection of the inputs and data frequency introduced in the ANN. The advantages of using just a sample of the set of data, is that the computational time is significantly reduced and more time can be used trying a higher diversity of models and combination of hyperparameters for the network. Although it is actually part of the building process, due to the large volume of work done during this section, it was considered to present it separately from the previous section.

First of all, the precise data for this test was selected as follows; the consumption over the years is slightly changing and for the two last years the distribution, mean and std, were quite similar. This added to the technological improvements it is thinkable that consumption will continue changing. Yet from a year to the following one this change will not be to significant. Taking all this into account, the sample of the last available year's data was selected for the development set, which represents a 25 % of the whole set of data.

After according the values of the hyperparameters of study, to optimize the forecasting of the network,***Section 5***, was time to select the different models to fed the network with.

6.1 Categorical variables

Is interesting to point out briefly, before creating the models, what a categorical variable is and mention the common ways to introduce them as inputs. Categorical data are variables that contain label values rather than numeric values, where each value represents a different category. The issue is that many machine learning algorithms cannot operate on label data directly. They require all input and output variables to be numeric. Hence, there are many ways to handle categorical data with neural networks [Ked17]. The most commonly used ones are the Ordinal encoding; an integer is assigned to each category. It does not add any new columns to the data, but implies an order to the variable that may not actually exist. And the One Hot Encoding (OHE); transforms a single variable with n observations and d distinct values, to d binary variables with n observations individually. Each observation indicating the presence (1) or absence (0) of the dichotomous binary variable.

Both procedures have some disadvantages, ordinal encoding would assume the existence of some hierarchy in the categories, as if February would be 'better' or 'higher' than January. While OHE increase the number of inputs and thus the complexity of the

net. However the second one was used to describe the models, then the hierarchical bias was removed from the training.

6.2 Models

In a previous part of the project, ***Section 4***, an exploratory analysis of the data was performed aiming to discover any kind of relevant insights on the set such as the correlations, trends, etc. Is during the creation of the different models where this information was first applied.

The aim of this project is the study and performance of developing an ANN capable of predicting, with the highest accuracy as possible, the demand in electric energy of a particular dwelling a day in ahead.

Model 1;

- Identification of the weeks day
- Value of consumption at same precise moment, on the previous day
- Value of consumption at same precise moment,two days before

Total number of Inputs = 9

Model 2;

- Identification of the weeks day
- Value of consumption at same precise moment, on the previous day
- Value of consumption at same precise moment,two days before
- Binary input representing either it is a festivity day or not

Total number of Inputs = 10

Model 3;

- Identification of the weeks day
- Value of consumption at same precise moment, on the previous day
- Value of consumption at same precise moment,two days before
- Season of the year

Total number of Inputs = 13

Model 4;

- Season of the year
- Value of consumption at same precise moment, on the previous day
- Value of consumption at same precise moment,two days before

Total number of Inputs = 6

Model 5;

- Values of consumption from 1h until 12h before, in a time-frequency of an hour

Total number of Inputs = 12

Model 6;

- Values of consumption at same precise moment, from 1 until 12 days before

Total number of Inputs = 12

Model 7;

- Values of consumption from 1h, 2h, 4h, 6h, 8h, 10h and 12h before
- Value of consumption at same precise moment, on the previous day
- Identification of the weeks day[22]

Total number of Inputs = 12

Not all models were done at the same point. Indeed, 1st to 4th model were done firstly in order to select the significant inputs and discard the less relevant ones, basically regarding the correlations and trends studied during the EDA. Secondly, 5th and 6th model were created to compare which proximity is more informative and, therefore, serves better to the training of the network. Finally, the 7th model was created as a result of the previous analysis. Source code of the model creation procedure at *Annex A I*.

6.3 Results

When thinking of how to evaluate the performance of an ANN there several aspects to study. Consequently a coefficient was created to fairly compare the different models. This coefficient (α), is composed of three variables; accuracy, computational time and complexity.

Obviously the accuracy, how the network is capable of approximating the output to the real target values, is the most important aspect. An accuracy measure is often defined in terms of the forecasting error which is the difference between the real and the predicted value. There are several measures of accuracy each with some advantages and limitations. For this particular problem the MAPE[23] was selected. Between complexity and time, complexity received more relevance. Being, in case of a real implementation, the memory required to perform the computation and forecasting dependant on the

[22]For this model a slight change was done; instead of having a binary value for each of the days, the number of inputs was reduced creating a category for the Monday, Tuesday, Wednesday and Thursday together. An individual one was done for each of the other days.

[23]Mean Absolute Percentage Error.

complexity of the net. The complexity was described as the number of inputs, considering that the more inputs the more connections and so complexity the network has. Lastly the modeling time was also added to the formula with lowest weight. In order to compare times, the computational time required to perform all combinations of hyperparameters was recorded in minutes.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{\|y_i - \hat{y}_i\|}{y_i} \tag{6.1}$$

Where y is the target value, $\hat{y}$ is the predicted value and N represents the total number of samples the network is trained with.

$$\alpha^{24} = 0.55 * MAPE + 0.35 * inputs + 0.1 * time \tag{6.2}$$

Once the first six models were done and the possible values for the hyperparameters were selected, was time to start with the development test. Therefore, with several for-loops, all combinations of parameters were computed for each model. Afterwards, the accuracy of last epochs was selected and the five best of them were saved together with their respective combinations. Finally with the mentioned values of the MAPE their mean was computed and used to compare the models.

The ***Table 6.1*** is the result of the previously explained selection for the case of model 7 with the ReLU function and Adagrad optimizer. Therefore, at ***Table 6.3*** at the corresponding cell, is found the mean of this MAPE values, right at the left of the time that took to compute the simulations of Model 7 for the 108 possible combinations of hyperparameters.

	Combination	MAPE
82	1_3_1_3_3	30.322
86	1_3_2_1_3	30.390
80	1_3_1_3_1	30.392
90	1_3_2_2_3	30.415
92	1_3_2_3_1	30.416

Table 6.1: Best five MAPEs of Model 7.

[24] Definition of the Coefficient.

	ReLU SGD		Tanh SGD		ReLU Adagrad	
Model	MAPE(%)	Time(min)	MAPE(%)	Time(min)	MAPE(%)	Time(min)
1	40.886	35.13	40.884	34.85	40.889	37.77
2	42.885	44.85	42.181	49.79	40.382	43.79
3	41.084	44.04	45.019	41.50	39.690	44.74
4	43.412	36.83	44.037	36.48	42.996	38.23
5	35.142	45.75	37.650	43.72	30.853	41.69
6	39.518	35.76	39.713	37.63	37.685	43.57

Table 6.2: Results of first models

Focusing on the hyperparameters, firstly the ones related with the training algorithm, the Adagrad showed a better performance. Regarding the activation function, although ReLU seemed to perform better, there is not a clear difference. Moving to the models, ***Table 6.2*** shows a visible difference in MAPE, however as commented in ***Section 5.3*** the neural network is initialised with random values of gains or weights, thus resulted in different starting points per each simulation during the training phase. Consequently, same model with equal structure and hyperparameters may result in a range of possible performance or accuracy values. Therefore an experiment , ***Subsection*** 6.3.1, was done in order to find out this range of values, which determine the difference in performance that allow to dictate that one model is better or worst than the others.

From models 1 to 4, the worst one was the 4th, which appeared to be the only one without the day of the week as an input. Focusing on the three first ones, there was not a significant difference between their performances. Therefore the labels informing about the season of the year[25] and either if the target day was a public holiday or not[26], were not necessary for the network, as those features didn't appear to make a difference on performance. Finally, models 5 and 6 had best performance than the others, which is reasonable considering that all inputs they had were values of consumption from previous time-steps. Between them, the fifth model had better results, which 12 inputs represented the consumption that the sensor registered from the 12 previous hours to the target time.

Consequently, with this results model 7 was defined using a mixture of the inputs from the best performing models with some modification. For example the day of the week was identified being Monday, Tuesday, Wednesday and Thursday considered equals as they showed similar behaviour during the EDA. Likewise, the information about consumption of last 12 previous hours was introduced with lower frequency. All this modifications led to a better model without such a significant level of complexity.

[25] Model 2.
[26] Model 3.

	ReLU SGD		Tanh SGD		ReLU Adagrad	
Model	MAPE(%)	Time(min)	MAPE(%)	Time(min)	MAPE(%)	Time(min)
7	30.486	36.90	36.005	33.63	30.382	37.21

Table 6.3: Results of last models

Model	α
1	29.411
2	30.088
3	30.852
4	29.574
5	25.337
6	29.283
7	24.597

Table 6.4: Coefficient values.

Table 6.4 shows lowest coefficients for the last three models. Hence, models 5 to 7 were selected for the training test.

6.3.1 1000 iterations experiment

The purpose of this experiment was to determine the range of values of MAPE in which was considerably an equal performance. The randomness provided by the SO and the initialization of the network weights, determines that the same specific network on the same specific training data will feed a different network with a different model skill each time the training algorithm is executed. Therefore, same model with equal structure, and the Glorot Uniform initialization, was run a thousand times. Meanwhile the MAPE of each simulation was being recorded in a data frame. The following table and plots manifest the result of the experiment:

	mean of 10 best mape
count	1000.000000
mean	47.506672
std	0.523260
min	46.447601
25%	47.149375
50%	47.441861
75%	47.771280
max	50.029321

Table 6.5: Statistical description.

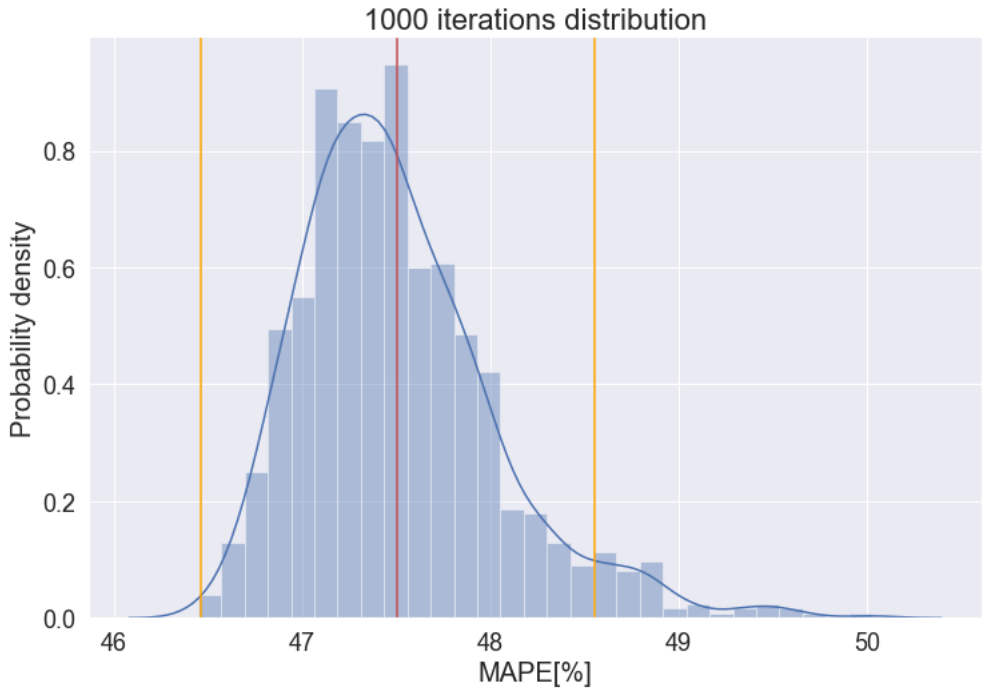


Figure 6.1: Distribution of the MAPE for 1000 simulations. The blue curve a density probability function, the total area under the curve integrates in to one. The y axis indicate the value of the probability density, while the x axis indicates the MAPE values. The vertical lines; the red one, signals the mean of the distribution while both yellow ones indicate the limits were the 95% of the values are found.

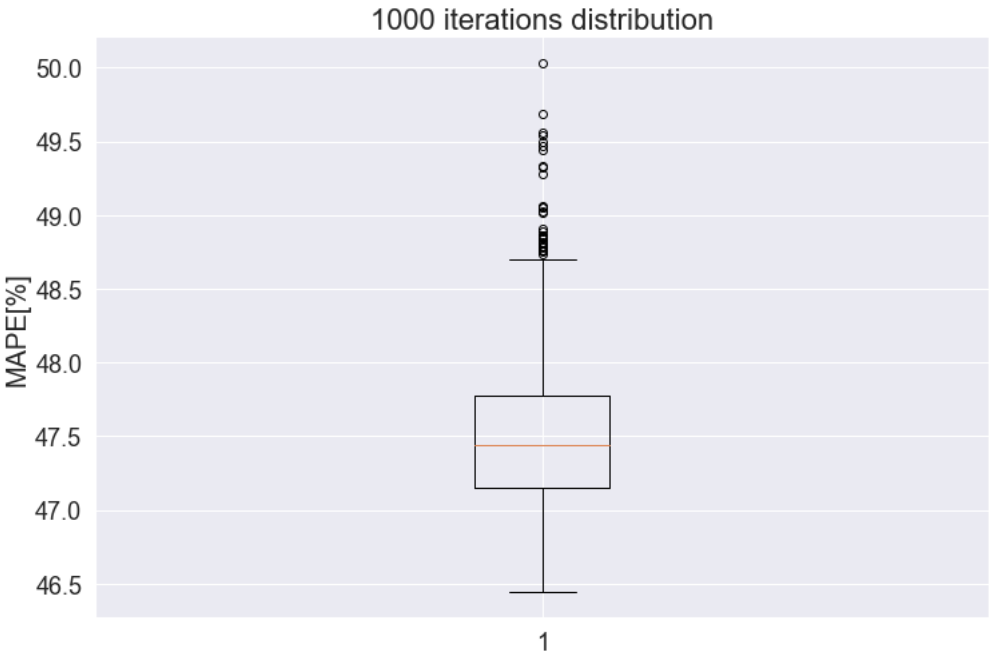


Figure 6.2: Box plot of the MAPE for 1000 simulations

The table represents a summary of the resulting distribution features. Both plots show

the range where the majority of the values, a 95%[27] of them, were found. Hence, was concluded that there was a difference of 2 percentage points in the MAPE value of the net which had to be considered as a margin when comparing models.

6.4 Hyperparameters

The optimal combination of hyperparameters was selected after choosing the models. From the best combinations of the selected models the number of hidden nodes and the learning rate were clearly 50 and 0.01 respectively. ***Fig. 6.3*** represents the accuracy of the 108 combinations performed with Model 7 using the ReLU function to activate the HL and Adagrad as optimizer algorithm. The way the combinations were codified led this plot to be divided by thirds, source code of the hyperparameter searching and codification at *Annex A II*. Consequently, the most significant observation, apart from the peaks[28], is the three clearly divided zones, due to the variation on the number of HN. First third equals to 12, the second to 24 and the last one to 50 HN. The instructions of the codification are found at the annex.

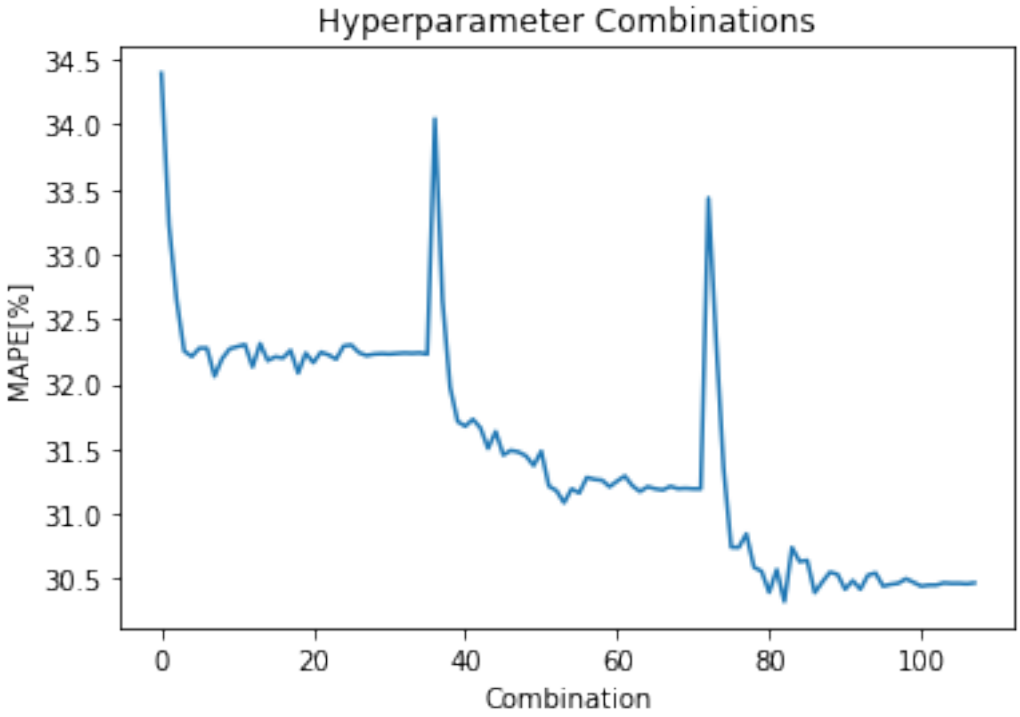


Figure 6.3: Combinations of Hyperparameters.

On the other hand, the optimal values for the batch size and the number of epochs were not well defined. Combinations with a batch size of 32 and 64 and 50, 100 and 150 epochs all appeared in best combination of hyperparmeters of some models without a clear trend. So a last experiment was done comparing those values, concluding with 64

[27] From the mean to two times the std on both sides, is holded a 95% of the data, marked by the orange vertical lines in ***Fig. 6.1***. Box plot shows same result.

[28] Peaks showed that worst thing was to combine a low number of HN, 12, with a lr of 0.05.

and 150 as the more suitable values for the problem, experiment description and results at *Annex B II*.

Hyperparameter	Value
Hidden Nodes	50
Hidden Layers	1
Activation of the HL	ReLU
Activation of the OL	Sigmoid
Weights initialization	Random Uniform
Learning Rate	0.01
Decay	1e-6
Optimization Algorithm	Adagrad
Batch Size	64
Epochs	150

Table 6.6: Hyperparameters selected for the training.

Those are the final values selected for each of the hyperparameters, used during the training of the ANN.

6.5 Frequency

Frequency is an aspect of the dataset which has been already mentioned in previous sections. During the EDA frequency was modified to show higher diversity of insights around the data set. However, still any analysis, related with the optimization of the ANN, has been presented.

After selecting the best models, combination of hyperparameters and structure for the net, three frequencies were compared. Thus, a training test was performed for models 5 to 7[29], with three different data frequences; 15 minutes, 30 minutes and hourly. The test resulted in a better performance of the network when it was fed with a frequency of an hour, experiment description and results at *Annex B I*.

[29] All models with the already selected values of each parameter.

7 Training process

In this chapter, the procedure related with the training of the ANN is exposed. Starting with the split of the set over Training and Test samples, the modifications done to the network parameters and closing the chapter sharing the results found applying the different models.

7.1 Training/Test split

As mentioned earlier, a training-test split is typically required for an ANN forecast. The test set has to follow two conditions: First of all, it must be large enough to yield statistically meaningful results. Secondly, the test set needs to be representative of the data set as a whole [Goo19]. In other words, this set of data must not contain different characteristics than the training set. Assuming that the test set meets the preceding two conditions, the goal is to create a model that generalizes well to new data.

The first issue here is the division of the data into both sets. Although there is no general solution to this problem, several factors such as the problem characteristics, the type of data and the size of the data set should be considered while making the decision. The literature offers little guidance in selecting the training and test samples. In general the sample size is closely related to the required accuracy of the problem. The larger the size, the more accurate the results will be. For this particular problem, the models and the selected network hyperparameters did not represent a high complexity, hence, the data size was not a limiting factor of the accuracy achieved. Therefore from a 70/30 to a 90/10 split there was not a big difference in performance[30].

In agreement with the precedent arguments, the following piece of code shows how training and test sets, for model 7, were split, 90/10, randomly using the *sample*() function from *Pandas*.

Code for training/test split:

```
m7_train = m7.sample(frac=0.9, random_state=1)
m7_test = m7.drop(i9train.index)
xtrain, ytrain = m7_train.iloc[:, :12], m7_train.iloc[:, -1:]
xtest, ytest = m7_test.iloc[:, :12], m7_test.iloc[:, -1:]
ytest.head(8)
```

[30] 70/30 showed a 30.57% MAPE at test Vs. a 29.47% on 90/10.

Where $m7$ was a data frame containing the inputs of model 7 on the first columns and the target value on the last column.

Timestamp	Consumption
2015-01-03 14:00:00	0.439863
2015-01-03 15:00:00	0.214725
2015-01-03 18:00:00	0.136851
2015-01-03 20:00:00	0.173955
2015-01-04 03:00:00	0.178620
2015-01-04 06:00:00	0.132469
2015-01-04 17:00:00	0.135767
2015-01-04 18:00:00	0.178987

Table 7.1: First target values of the Test set.

Table 7.1 is the representation of the 8 first target values of the Test set.

As *Keras* allows, during the training, a validation split was done too. The model set apart a fraction of the training data, which did not train on, and evaluated the loss and metrics on this data at the end of each epoch. Source code of the training and test at *Annex A III*.

Is of high importance to never train on test data. Otherwise, it would not longer be accurately measuring how well the model generalizes to new data. A good indicator of training data on the test set is a much higher precision on the test than the training.

7.2 Modifications

On ***Section*** 6.4 the corresponding parameters of the network were properly selected. However, those decisions were not done with the whole set of date, so was assumable that some changes could be done during training. Consequently, aiming to optimize the forecasting, the initialization of the parameters was changed. Initially a Glorot Uniform initialization was selected, which draws samples from a uniform distribution where limits are ± 0.5. Another initializer was used during training which presented lower limits, ± 0.05. Likewise, the number of epochs was increased to 250 and the number of HN to 75. The corresponding increase of the accuracy due to the precedent modifications, is represented in the results.

7.3 Results

Same procedure was performed for the three models. After dividing the data in the two sets and preparing the network for the training and corresponding test, the algorithm was compiled. Next, the predictions were done with the test sample. This section is a summary of the results from the just mentioned steps and the conclusions based on them.

Firstly, after the training was performed the learning rates were plotted. This visualization serves as indicator of an acceptable number of epochs.

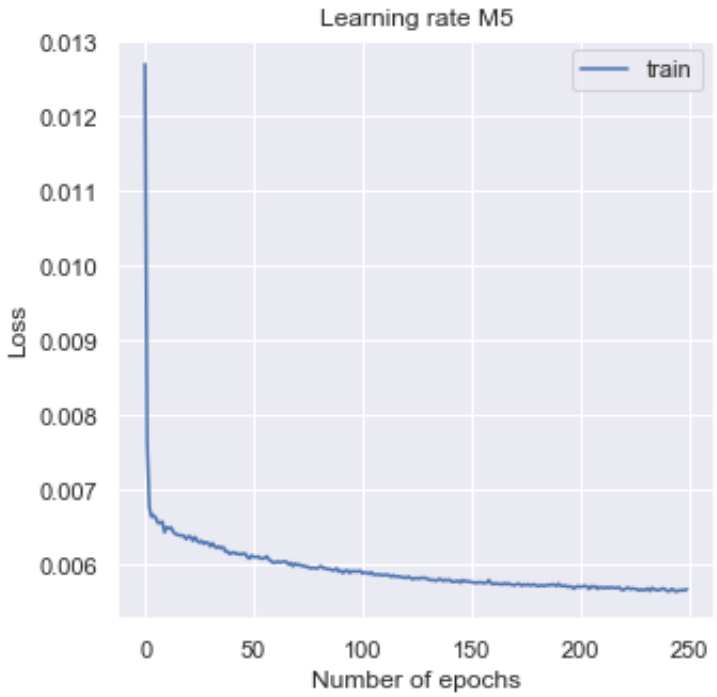


Figure 7.1: Learning Rate during the training of model 5.

At ***Fig.*** 7.1 the evolution of the loss function's value over the epochs for model 5 was represented. In other words, it shows how the performance of the network is improving during the training. It is perceptible how the function continued decreasing after 150 epochs till the end and how it would probably continue decreasing a bit more. However, the rate of improvement over time consumed and complexity was not worth it. Consequently, 250 epochs was chosen. For model 6 and 7 the learning rate was quite the same.

The following table contains the metric values and computing time from each model:

Model	MAPE[%]	Time[s]
5	28.601	114.72
6	36.476	108.02
7	24.977	110.81

Table 7.2: Training performance.

There was an improvement in performance of each model, compared to the development performance. A consequence of the modifications done to the network hyperparameters, and to the increase of the size of the data set used to fed the ANN. Regarding the time, is less than 2 minutes per training and quite similar between them. Considering that the complexity of the models was practically the same, they had same structure and similar inputs, similar times were expected. As a whole this results proved that is possible to get a better accuracy of a network by selecting the appropriate inputs, without this leading to an increase in complexity and time computing.

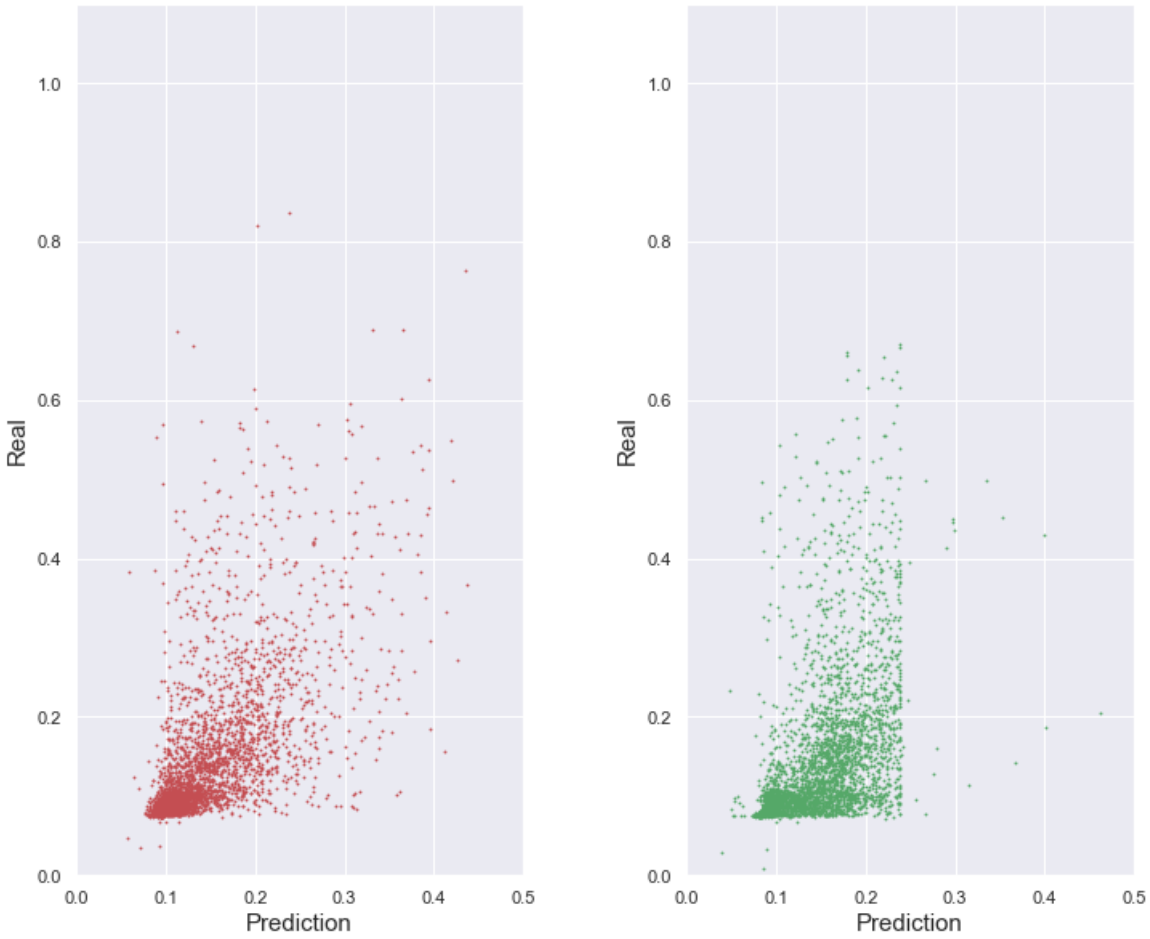
The results of the predictions for the Test data points was quantified numerically and

visually. The MAPE of the predictions, ***Table*** 7.3, were similar to the training ones while the time needed, for the forecasting of the 10% of the data set, was of 12 seconds.

Model	MAPE[%]	Time[s]
5	27.739	12
6	33.623	12
7	26.097	12

Table 7.3: Test performance

Next images represent the relationships between the real values and the predicted ones. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole. Therefore, in a forecasting problem this type of plot visually indicates how close are the predictions to the actual values. A result with 100% of accuracy would draw a perfect diagonal, with slope equals one, where each point would have same value for both axis.



(**a**) Model 5.

(**b**) Model 6.

Figure 7.2: Scatter plot of the predicted Vs the real values.

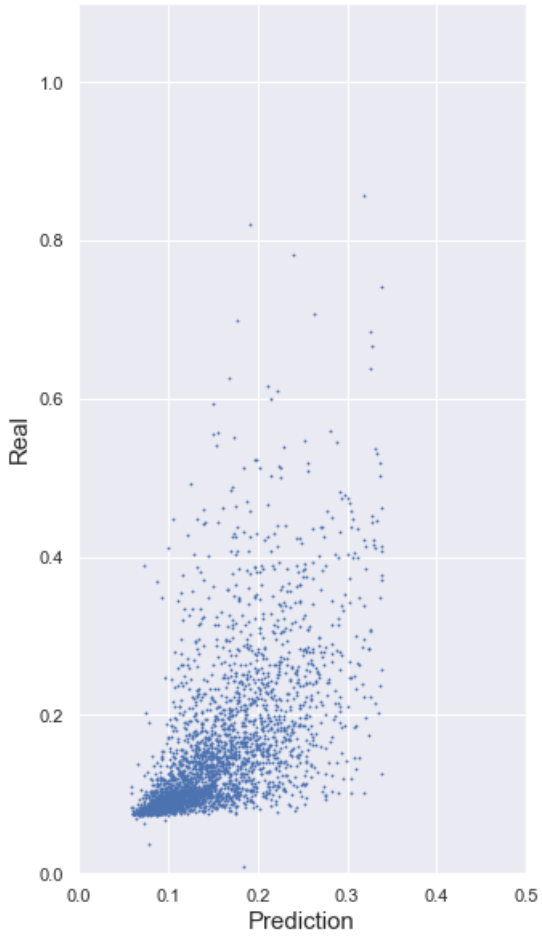


Figure 7.3: Scatter plot of the predicted Vs the real values, Model 7.

The data used for the representation of the plot was still scaled so values go from 0 to 1. The scatter plots show a higher density of points at lower values. This values correspond to the bast majority of the data. While less points are found with higher values, which correspond to the outliers. The shape represented by the dots show how for the central tendency, the difference between the actual values and the predictions is lower, in some cases there is no difference. However for the outliers, the network was not able to predict with certainty on high values. Instead, computed more values inside the central tendency range. Therefore, while the real sample of values was found from almost 0.1 to 0.8, the predicted sample was found from same lower threshold to little more than 0.4 for models 5 and 6, and around 0.35 on model 7. There is a phenomena, which created some kind of limits on the forecasted values. This effect draws a line on the plot, clearly seen in Model 6, also perceptible on Model 7.

Finally, the normalization of both the predicted and the actual set was inverted and some plots were generated in order to compare both samples. The aim was to visually represent the accuracy of the forecasting. This images show how prediction behaves on the central tendency as well with the peaks of consumption. Before starting the comparison, remember that test values were selected randomly so any kind of seasonality or trend is expected. The samples were reduced as the whole set was to big to clearly visualize the differences on both types of data. Therefore, as first comparison a subsample with 1000 points, 400 points and 200 points were plotted for each model. The

three plots for each model correspond to different data so that different situations were visualized in each image.

Model 5:

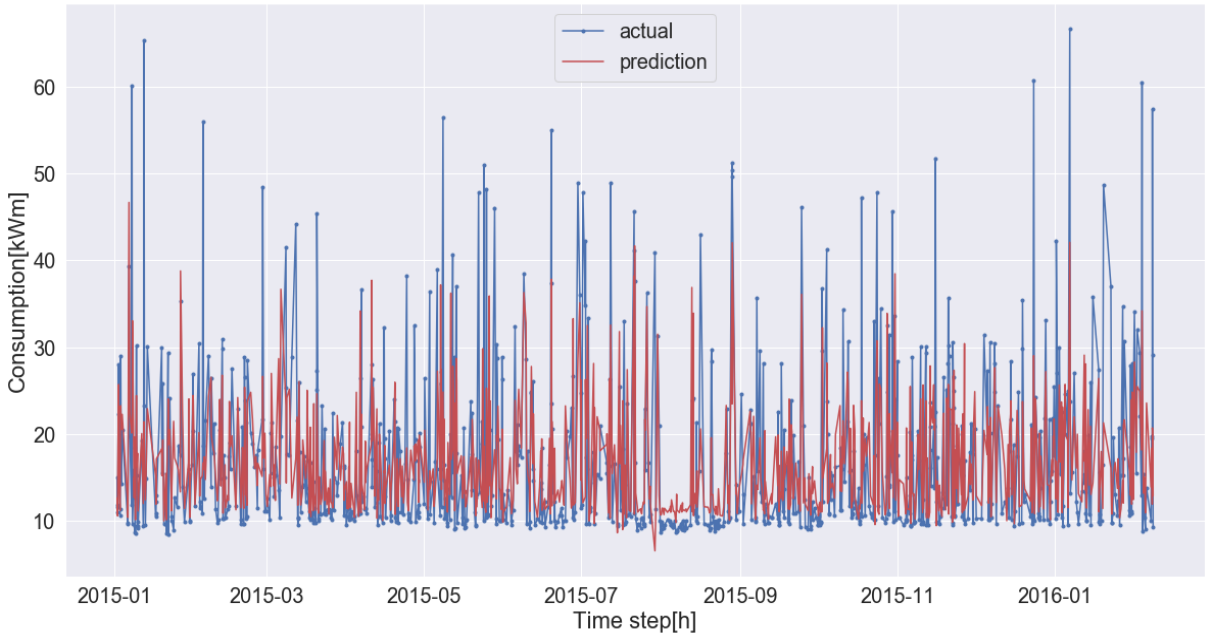


Figure 7.4: Forecasted Vs Real values, 1000 points.

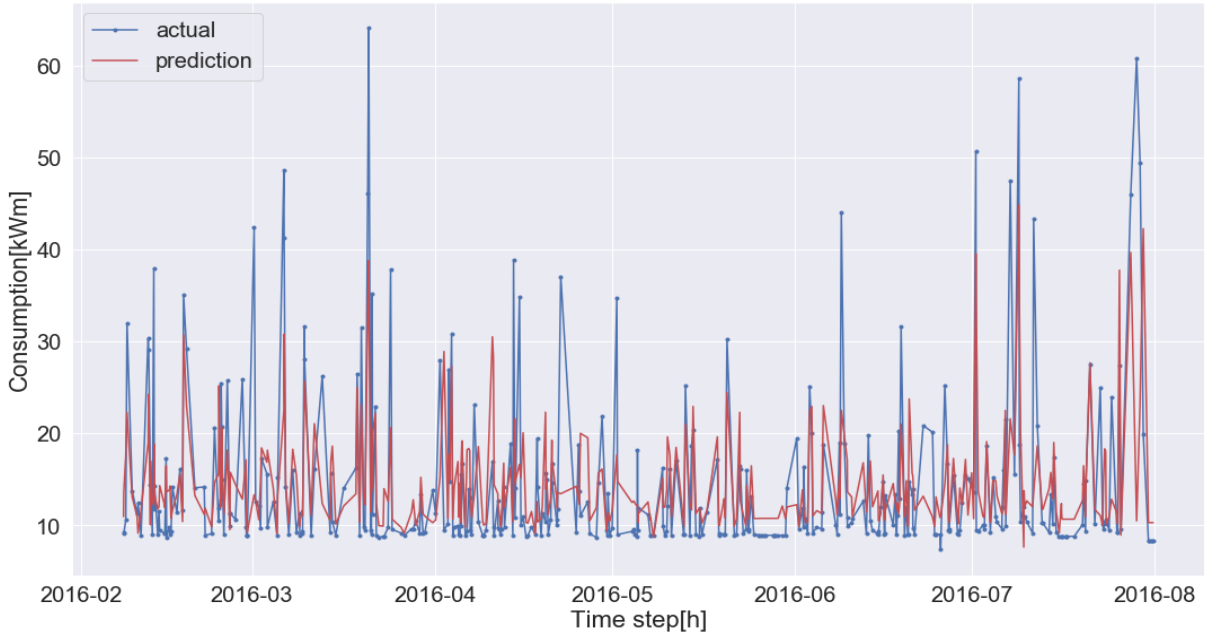


Figure 7.5: Forecasted Vs Real values, 400 points.

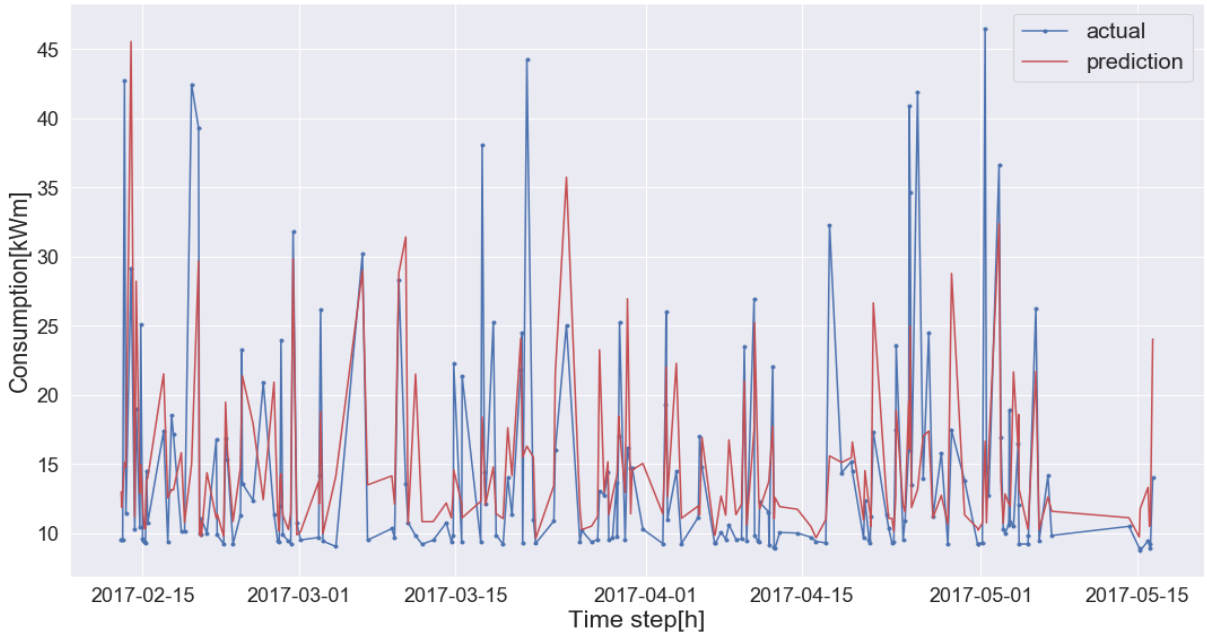


Figure 7.6: Forecasted Vs Real values, 200 points.

Model 6:

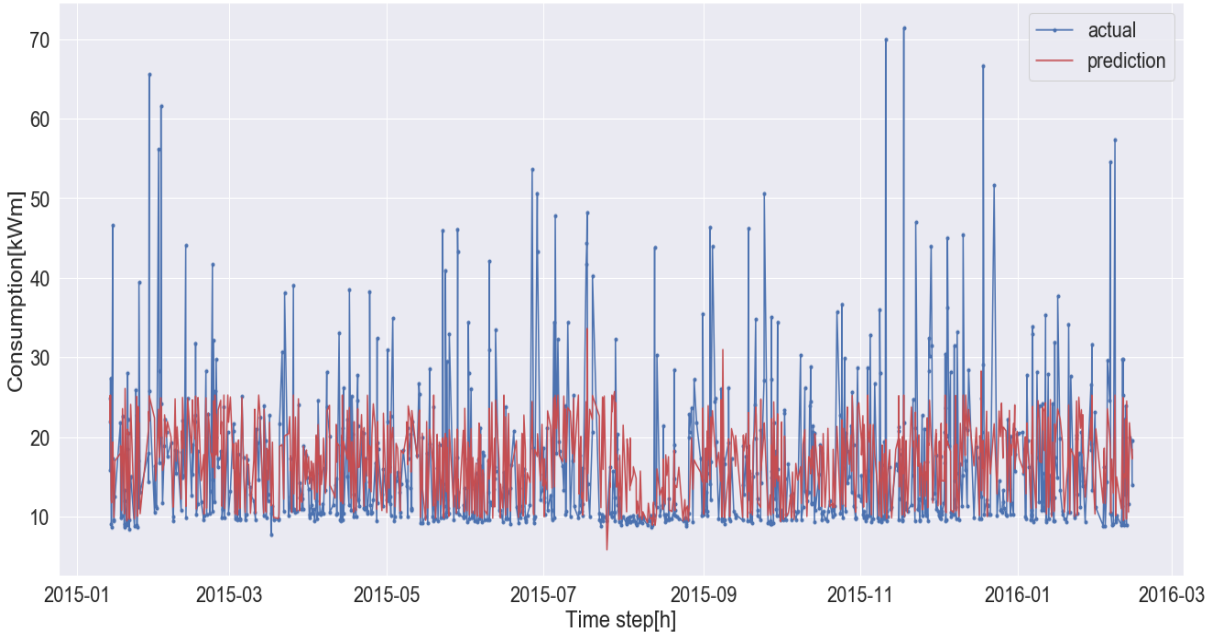


Figure 7.7: Forecasted Vs Real values, 1000 points.

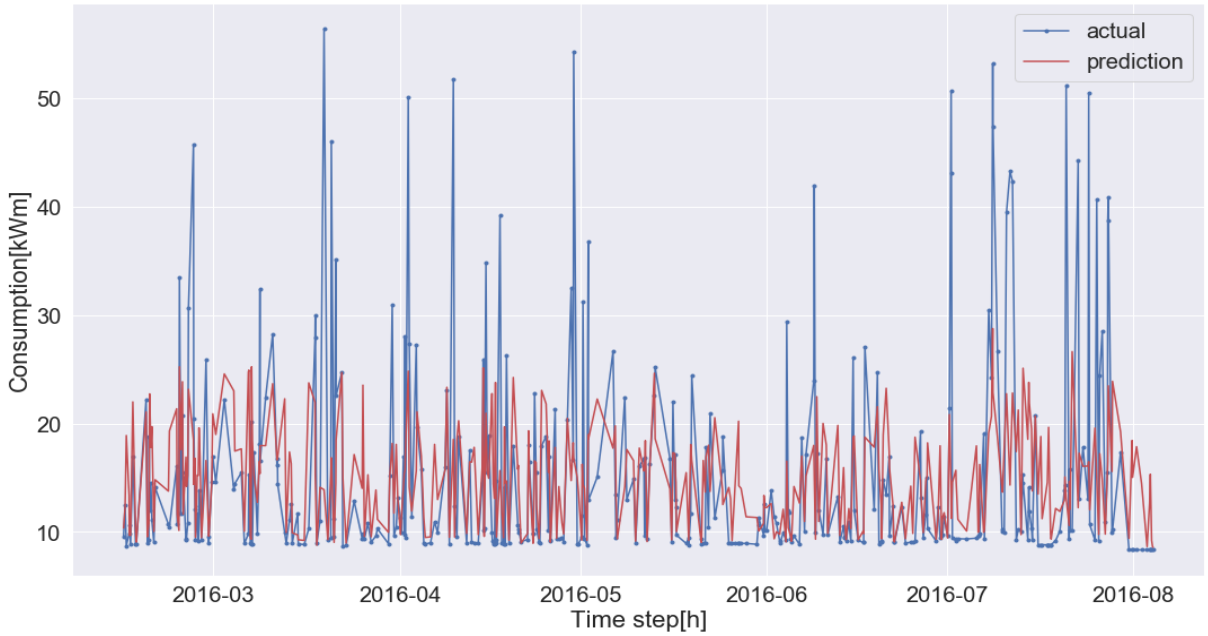


Figure 7.8: Forecasted Vs Real values, 400 points.

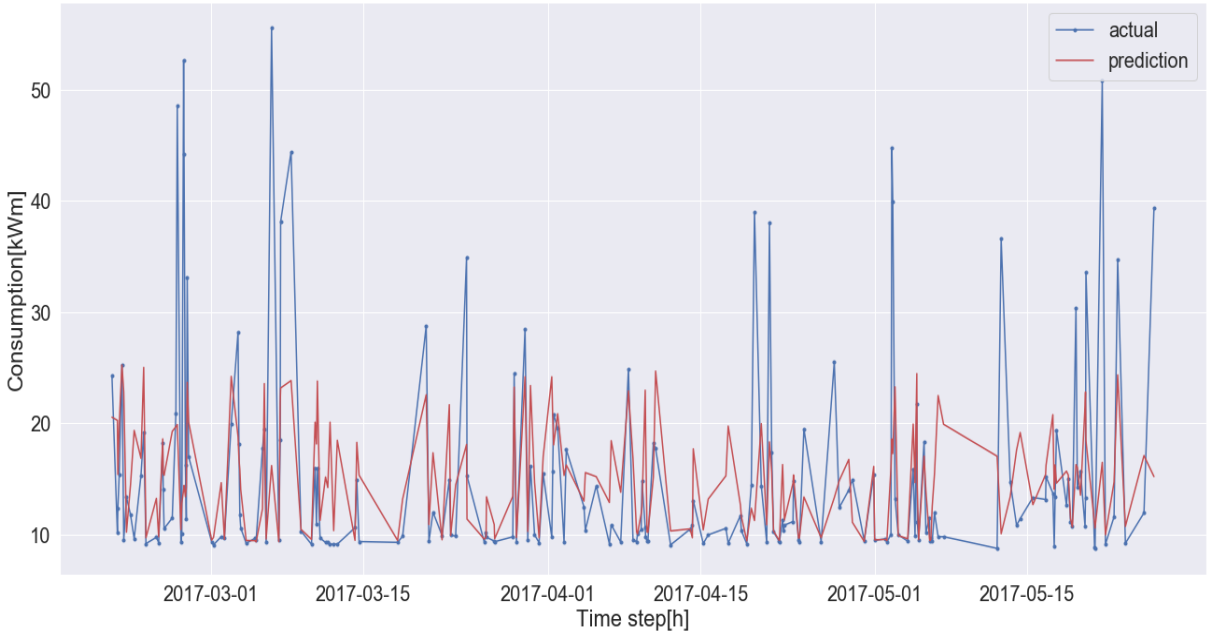


Figure 7.9: Forecasted Vs Real values, 200 points.

Model 7:

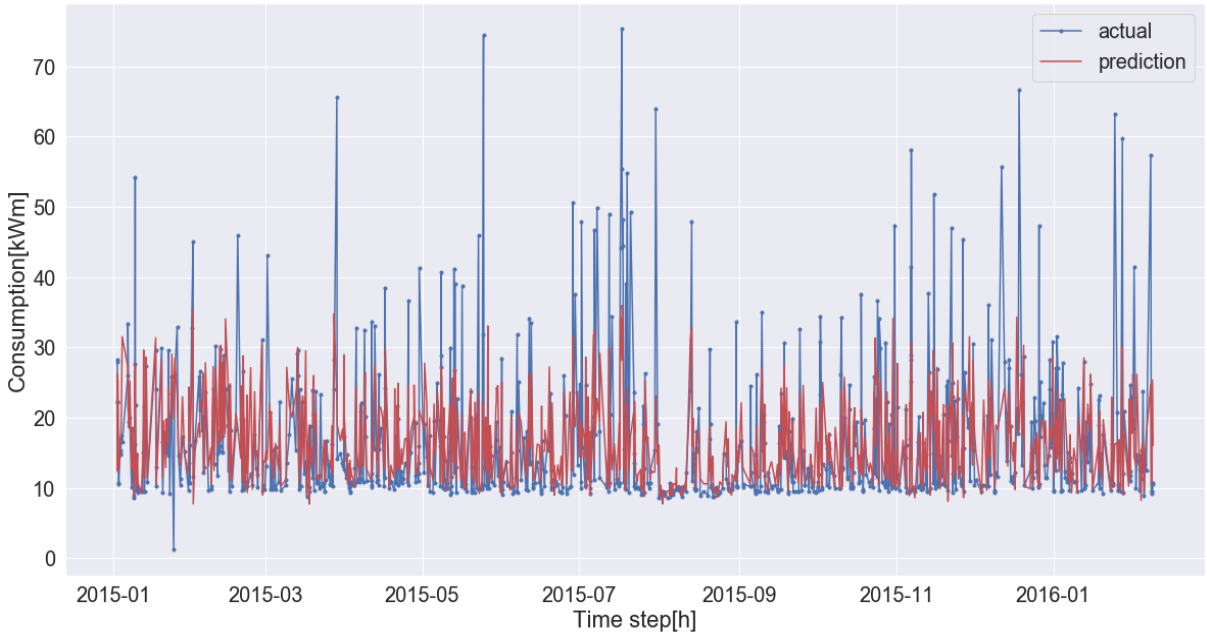


Figure 7.10: Forecasted Vs Real values, 1000 points.

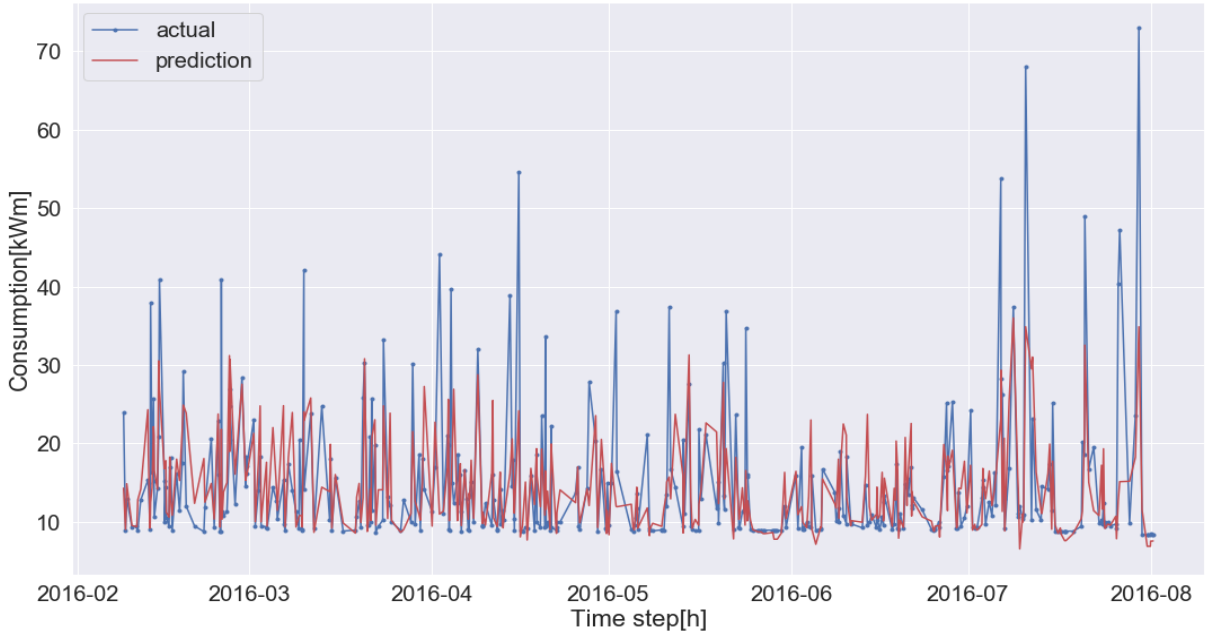


Figure 7.11: Forecasted Vs Real values, 400 points.

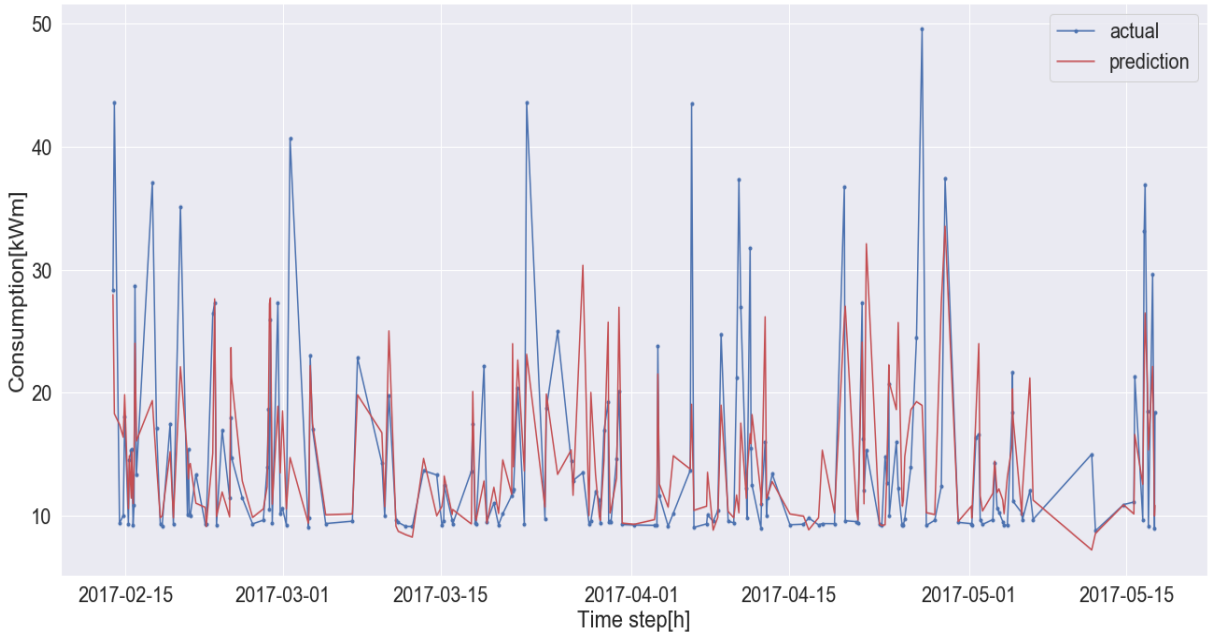


Figure 7.12: Forecasted Vs Real values, 200 points.

From each plot different aspects of the forecasting were presented. The visualization with the 1000 points show how the network predicted inside a range of values, lower than the real range of values. This interval was basically the central tendency of the actual values, and although forecasted values achieved some peaks as well, they were not the highest outliers. While some real values got higher than 60,000 Wm the predicted ones did not get higher than 48,000 Wm. However, in the 200-points plot is shown how the predictions seemed to follow the shape of the real values. Same result was seen with the 400 points. Although the accuracy of the network was not so good on higher values, it was better on lower ones. Together with the last type of plot they show with more detail how the forecasted points generally stayed close to the real points on middle and lower values. To sum up, taking into account the randomness of the points, the precedent images visualized how in general the network did a good job learning the general trends. Relating the plots with the results of the MAPE values; Model 6 was the one with significantly lower accuracy, which during the visualization showed a lower range of values compared with the other models.

7.4 Model 8

The results from the precedent models showed a fair accuracy for the forecasting of the electrical consumption of a particular dwelling. However, the way the inputs were selected for those models do not led to forecast over a whole day. Although using the values of consumption from the immediately previous hours to feed the network, as was thinkable, led to a higher accuracy, this fact was also the limiting factor of the model. In order to be able to predict for a whole day in advance, any value of consumption from the 23 last hours can not be fed into the network.

The aim of Model 8 was to perform the forecasting of electric energy demand of a whole day, 24 hours in advance. In addition, this model was used to compare the accuracy that the network reached when fed with the consumption values of the precedent hours, as

in Model 7, and without them.

Several models were created in order to select the proper model 8:

- Model 8.1; Information on the values of consumption of the day before the predicted one, 24 to 48 previous hours. It had identifiers of the day of the week and about the trimester of the year. Total number of inputs = 23.
- Model 8.2; Same values of consumption as in Model 8.1. However, no identifying labels were used. Total number of inputs = 15.
- Model 8.3; For this model just consumption from previous 24h to 36h was used. Total number of inputs = 9.
- Model 8.4; The inputs were the values of the consumption on the 24h, 36h and 48h previous hours to the last hour of the predicted day. Total number of inputs = 3
- Model 8.5; Same inputs as Model 8.4 adding the labels for the identification of the week's day. Total number of inputs = 7

Same process as in the first selection of models was done. Consequently, the development test was applied to the previous models. However, the MAPE was not enough to identify the optimal models as they resulted in extremely close values. Not a significant difference was found between them. Therefore the training was performed with all of them in order to identify the difference on their accuracy in other ways. Through the training, two of the hyperparamters from the network were changed for models 8 in order to boost the accuracy a bit more, the number of HN was increased to 100 and the number of epochs to 250. As was foreseeable, the metric value was again similar for the 5 models.

Model	**MAPE[%]**	**Time[s]**
8.1	38.024	174.68
8.2	38.714	162.54
8.3	39.988	150.07
8.4	40.002	128.96
8.5	38.604	128.38

Table 7.4: Training performance of Models 8

Despite of this similarity, the difference was found at the range of values each model was predicting at. Specifically, the more accurate a model was, a wider range of values did achieve. In contrast, the worse a model was in a more tightly closed range were their values. Therefore models with a higher range of values, approximated better the shape of the real consumption, more significantly on the outliers.

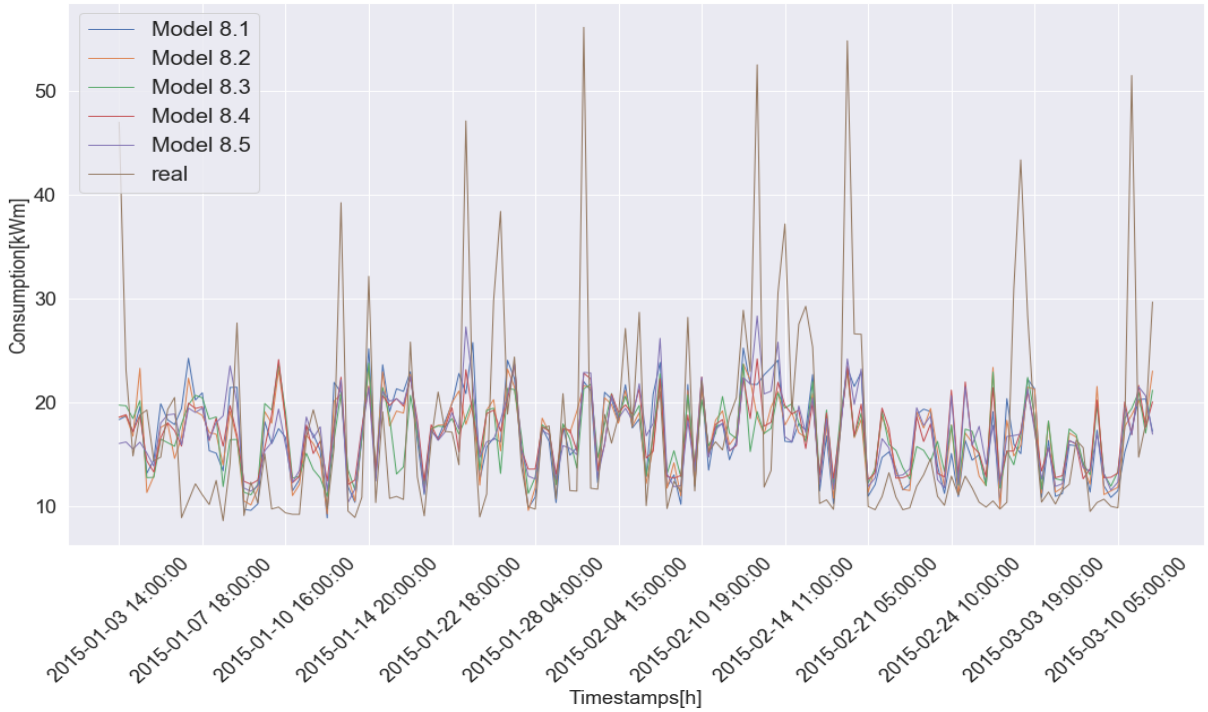


Figure 7.13: Real values Vs the 5 models, 150 points.

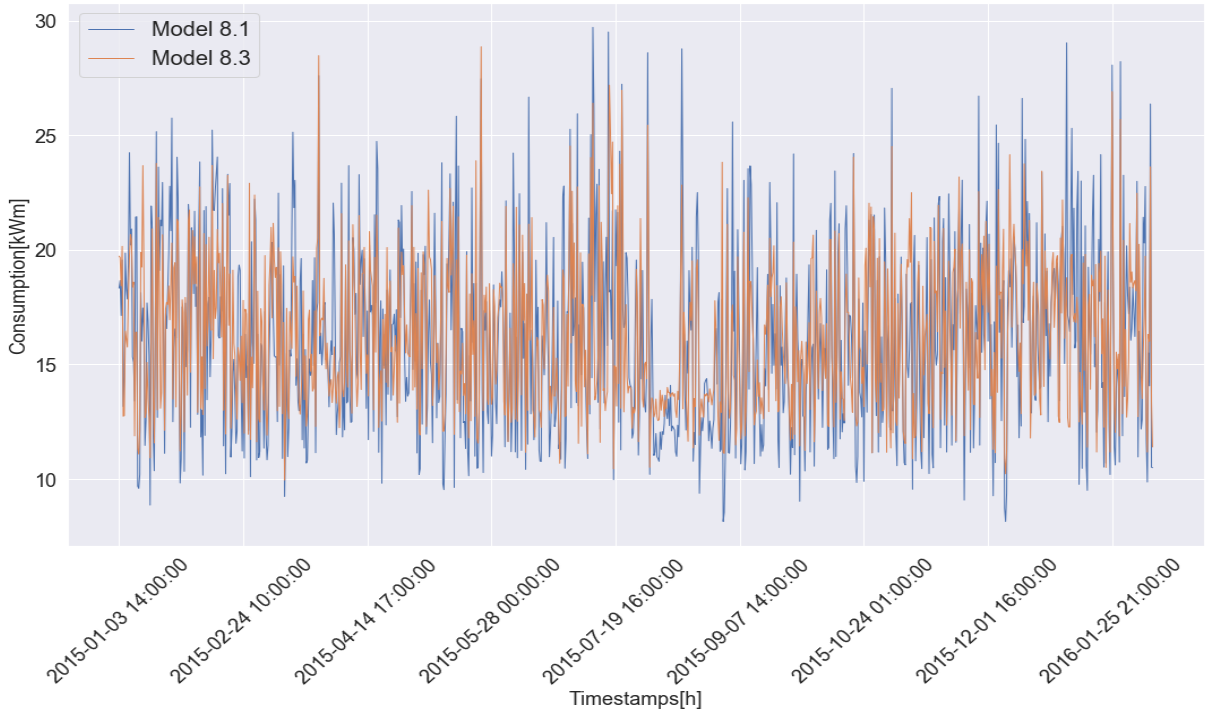


Figure 7.14: Model 8.1 Vs Model 8.3, 1000 points.

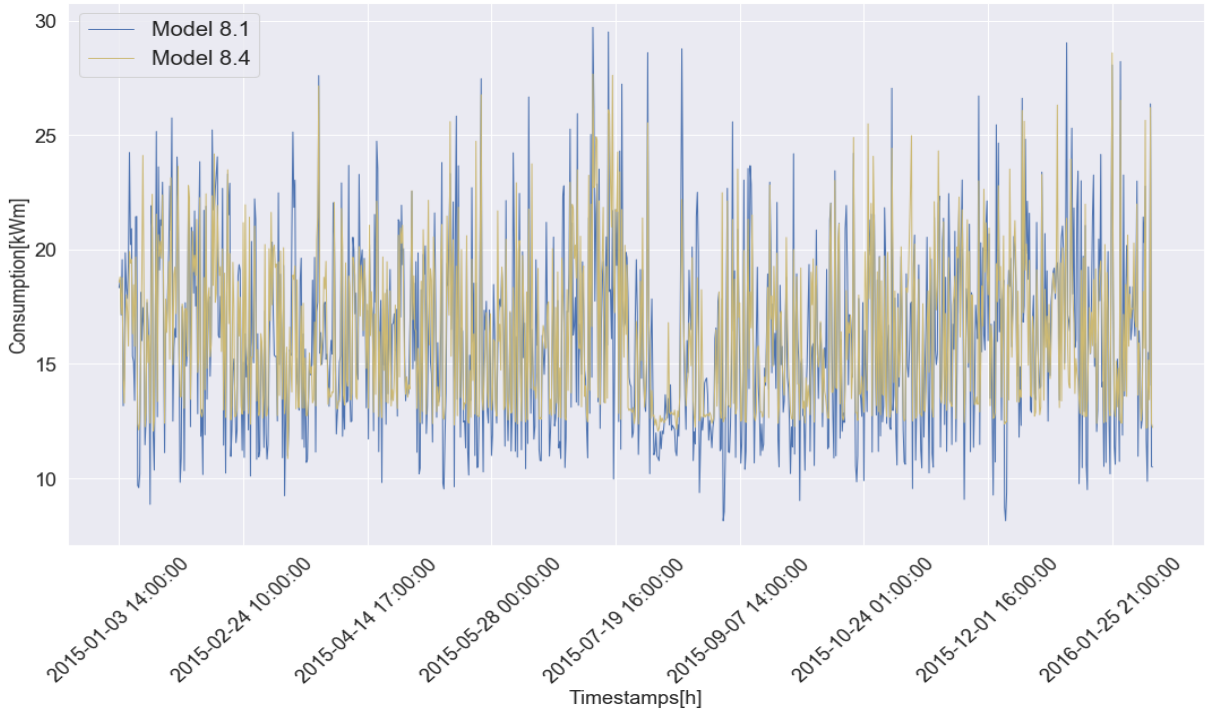


Figure 7.15: Model 8.1 Vs Model 8.3, 1000 points.

The actual sample of values had higher and lower points which the network was not able to achieve with the predictions as seen in ***Fig.***7.13 The model which was expected to have better performance results was model 8.1 while 8.3 and 8.4 were expected for the worst results. Therefore, in ***Fig.***7.14 and ***Fig.***7.15 a sample of the forecasted values of model 8.1 was plotted together with a sample of models 8.3 and 8.4 respectively. The images show how model 8.1 had points found above and below the range of values of the compared models. In the second image how model 8.1 achieved lower values better than 8.4 is more perceptible.

In order to clarify this fact, the following images describe the distribution of the values that models 8.1, 8.2, 8.3, 8.4 and 8.5 forecasted from the same test sample. Last figure is the distribution of the actual sample of consumption values.

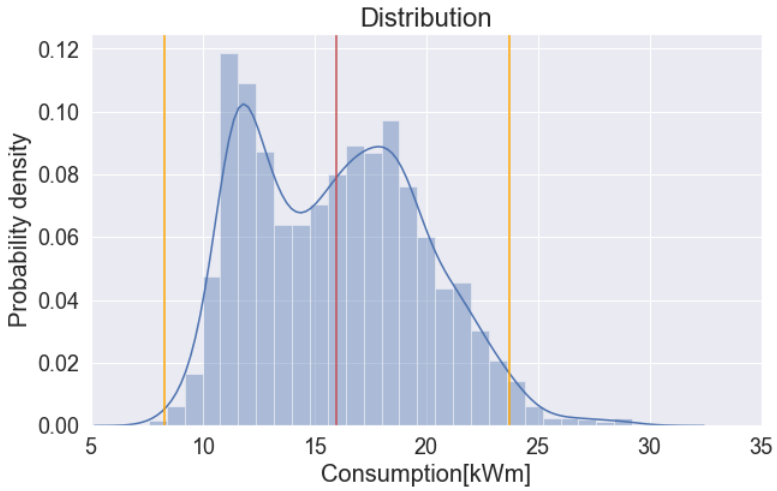


Figure 7.16: Distribution forecasted of values for Model 8.1. The blue curve is the density probability function, the total area under the curve integrates in to one. The y axis indicate the value of the probability density, while the x consumption values. The vertical lines; the red one, signals the mean of the distribution while both yellow ones indicate the limits were the 95% of the values are found.

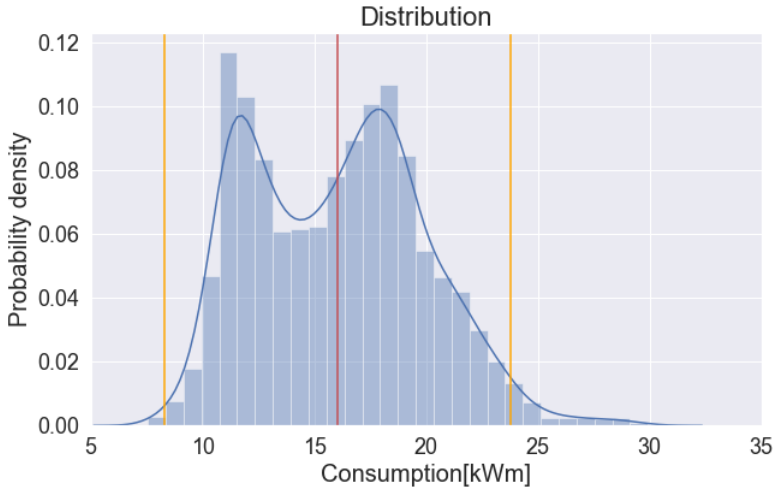


Figure 7.17: Distribution of forecasted values for Model 8.2.

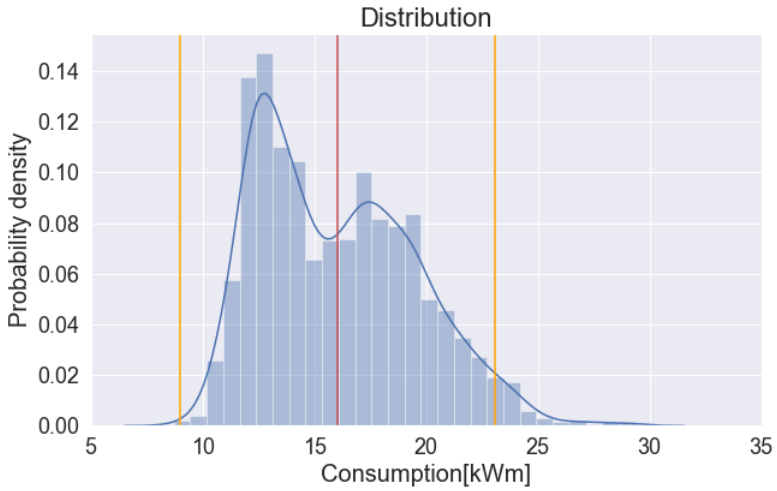


Figure 7.18: Distribution of forecasted values for Model 8.3.

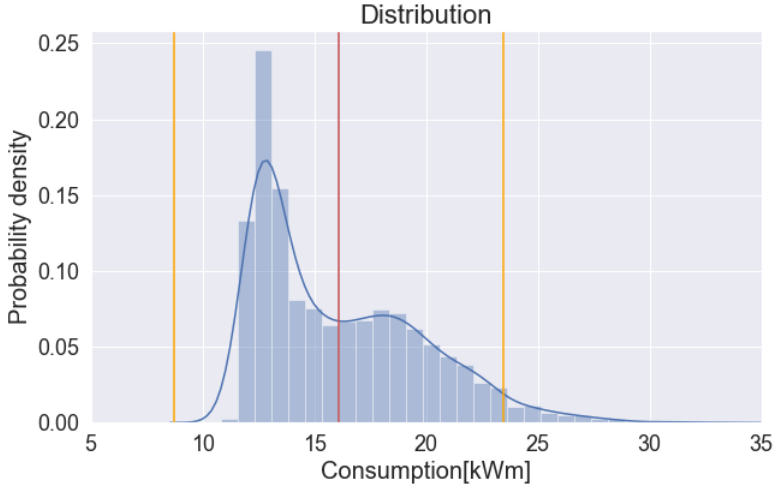


Figure 7.19: Distribution of forecasted values for Model 8.4.

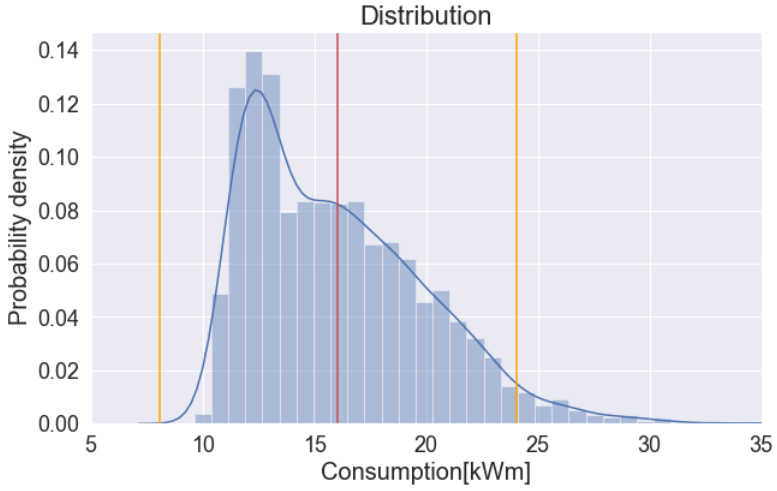


Figure 7.20: Distribution of forecasted values for Model 8.5.

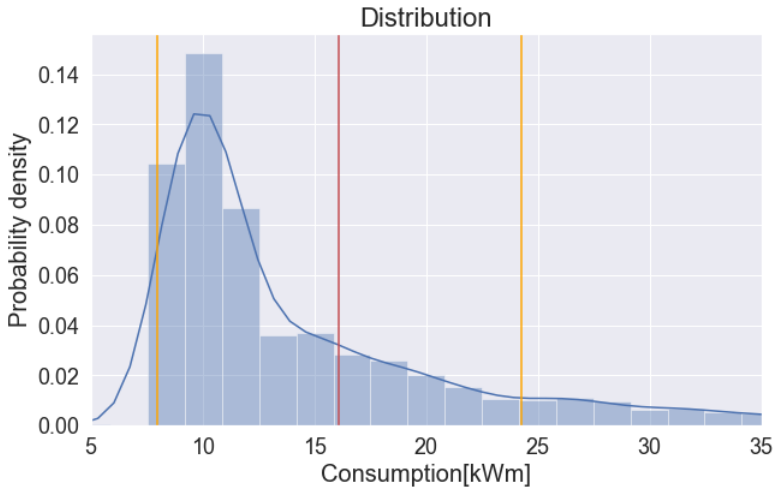


Figure 7.21: Distribution of the real values.

It can be seen how the sample of real values had an import amount of points at lower values, around the 10kWm. However, the results from the forecasting indicated that only Model 8.1 and 8.2 achieved to get an important amount of points, around 10-11,5kWm. In addition, ***Fig.*** 7.21 shows how some of the actual values were found over the 25kWm while models 8.3 and 8.4 barely had any point achieving that level. Better models such as 8.5 got some values around that point.

Consequently with this results, a mixture of models 8.1 and 8.5, which seemed to achieve better performances was created;

- Model 8; Values from the previous day in a frequency of 4h as consumption inputs. The indetifier for the day of the week was added too. Total number of inputs = 11

7.4.1 Results

The performance of Model 8 was qualified the same way as the other models previously trained.

Model	MAPE[%]	Time[s]
8	38.115	12

Table 7.5: Forecasting performance of Model 8

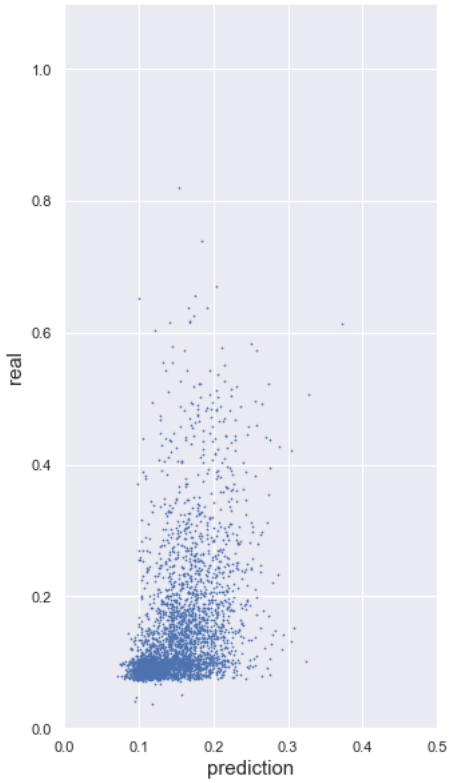


Figure 7.22: Scatter Plot: Actual values Vs the predicted values for Model 8.

Compared to Model 7, the scatter plot, ***Fig.??***, showed that the predicted values for Model 8 had less representation around the 0.3. While the other models still had a dense cloud of dots at x axis equal to 0.3, Model 8 barely had a few number of points at that level. Also over the left threshold, Model 7 achieved lower values.

The line plots visualizing random hours, comparing real and predicted values by Model 8 were also plotted.

Model 8:

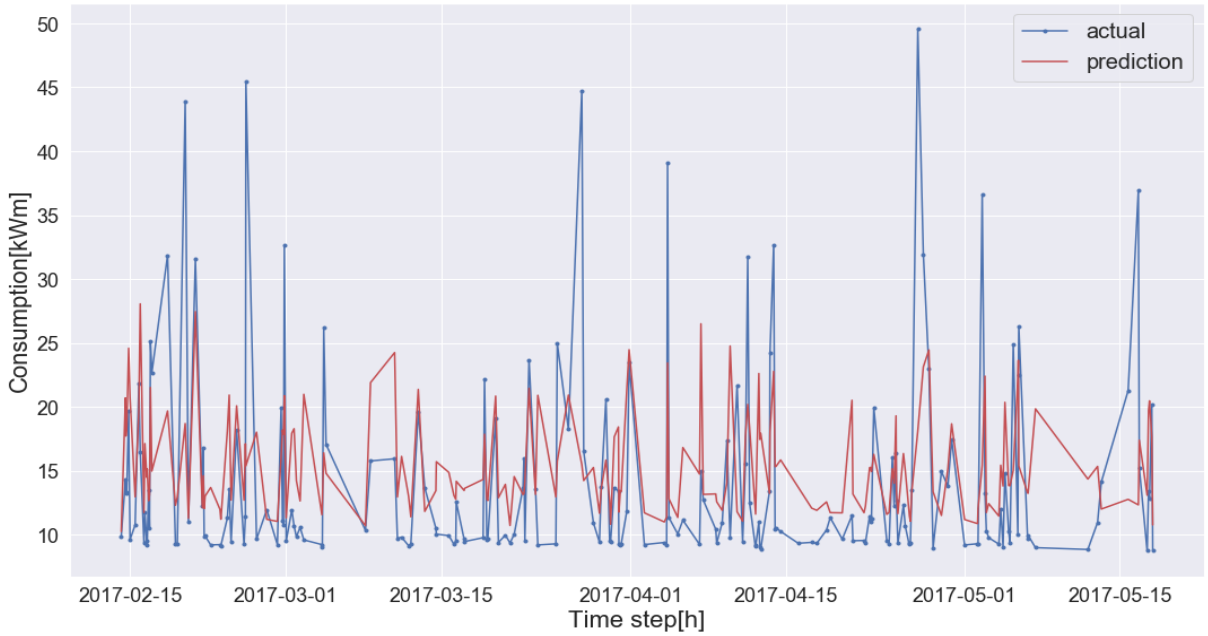


Figure 7.23: Forecasted Vs Real values, 200 points.

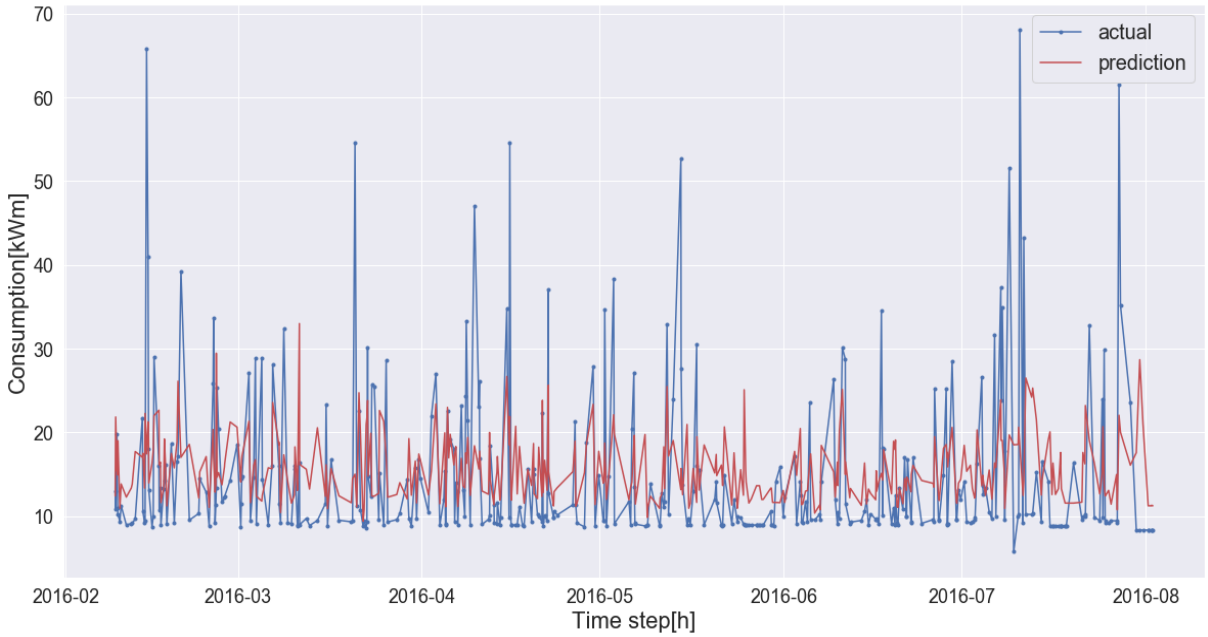


Figure 7.24: Forecasted Vs Real values, 400 points.

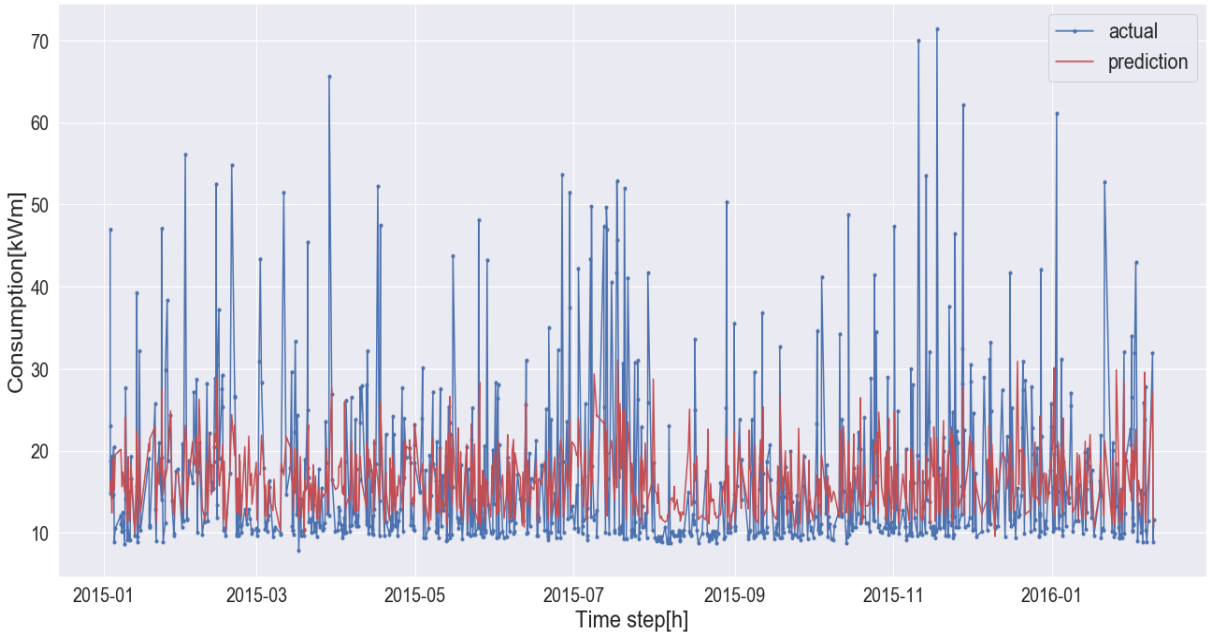


Figure 7.25: Forecasted Vs Real values, 1000 points.

Again, if comparing with the previous models the accuracy of Model 8 was weaker. Not only regarding the achieving of a narrowed range of values in the predictions of model 8, but also on the certainty over the central tendency where the forecasted values did not estimated the consumption as well as the previous models. This lost in accuracy was expected as the results presented in ***Section*** 7.3 were from models trained with values of electrical consumption from the immediate previous hours to the time of prediction. Consequently, was less aware of the behaviour of the immediate consumption.

8 Forecasting

This section is focused in the forecasting of different periods of time, using models 7 and 8. This periods were specifically selected for their characteristics. Thus, this section presents how the forecasting, of the mentioned models, behaved in diverse situations.

In order to carry out a deeper analysis on the performance of the forecasting of model 7 and 8, these models were subjected to: **a**) the forecasting of a period of time with low consumption, **b**) another one with high consumption and finally **c**) to the next day on the data set, first of January 2019.

May 2016:

During the EDA, ***Section*** 4, was observed that one of the months with lower consumption was May of the 2016. During this period of time, for a whole week the consumption dropped in stand by mode. Together with the first week of the month and three common days, these were the three periods of low consumption selected to perform the forecasting.

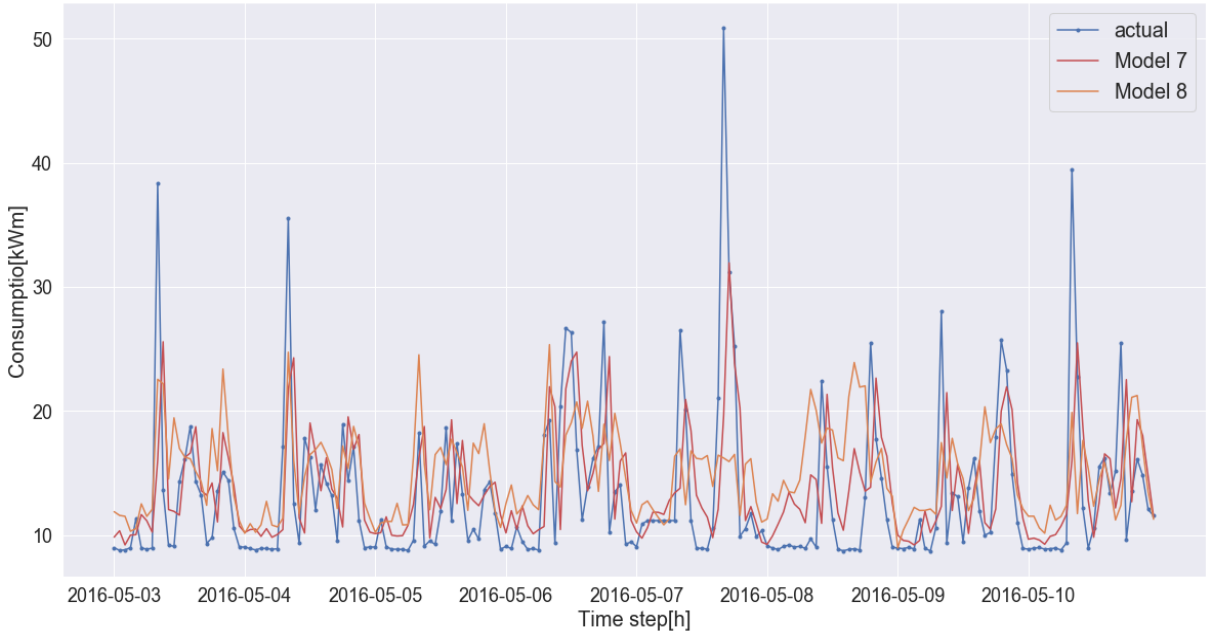


Figure 8.1: First week of May 2016, forecasted by Model 7 and 8 and compared to the actual values.

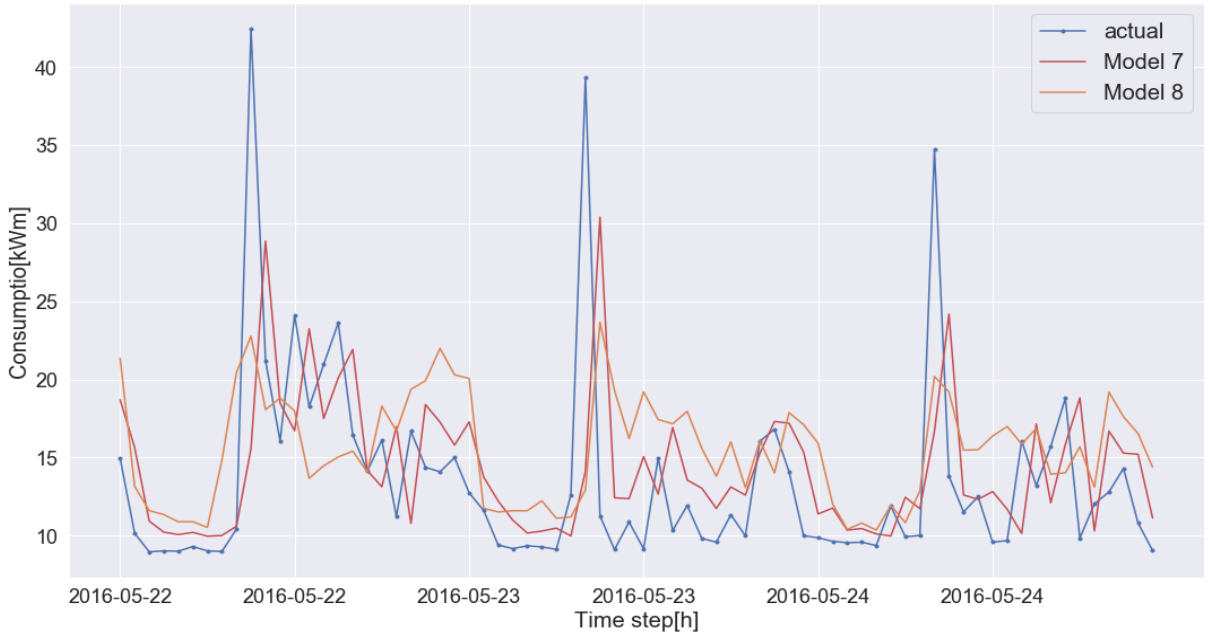


Figure 8.2: Days 22nd, 23rd and 24th of May 2016, forecasted by Model 7 and 8 and compared to the actual values.

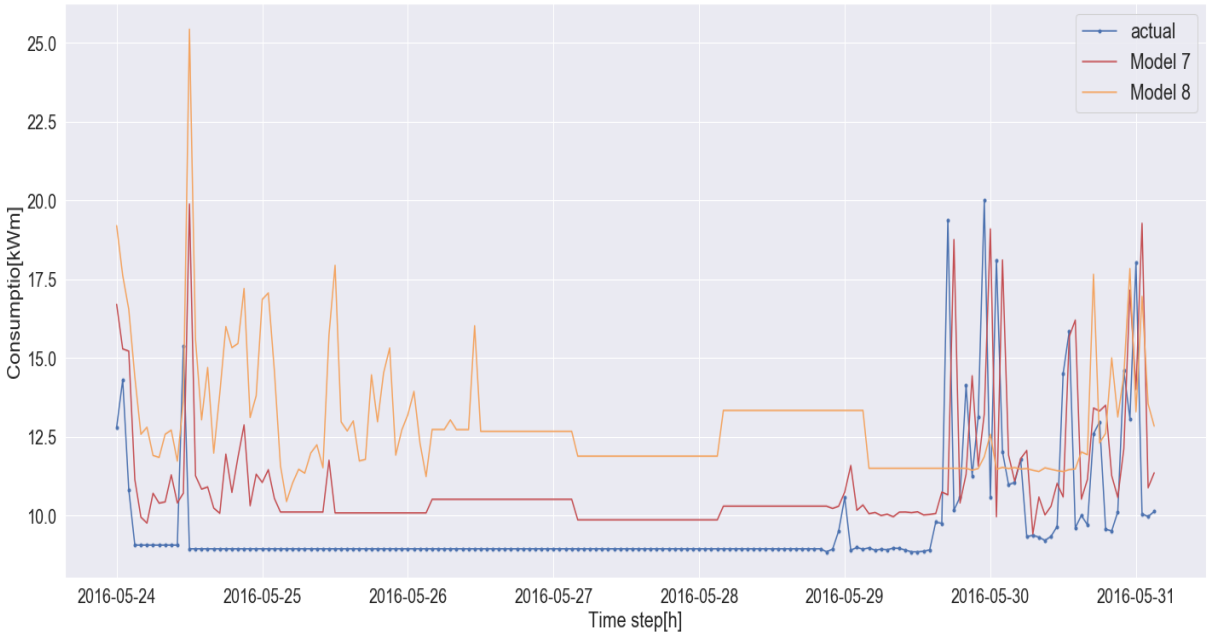


Figure 8.3: Drop on last week of May 2016, forecasted by Model 7 and 8 and compared to the actual values.

Overall, both models showed how, in general, they forecasted following the actual target shape of points. In addition, apart from the peaks, the points found in between 10kWm and 20kWm presented higher accuracy. From the daily trend is remarkable that, even though the demand of the household at sleeping-time is pretty similar for all the days, any of the models was able to predict those values with exactitude. In ***Fig.*** 8.2 is seen how Model 7 was more accurate, it even followed the exact same shape of the real values during some intervals of time. However it seemed to have a lag-hour most

of the time. The last image represents the drop in consumption, an anomaly of the time series, where was visualized how models were not able to adapt to the value level but after a time, which was longer in the case of Model 8, they achieved to forecast a low constant value as the real consumption did. In fact, Model 7 reacted 1 hour late to the ending of the constant period, when it started forecasting with high accuracy, but still one hour later. Model 8 took longer to react, actually a bit more than a day. As though the time of reaction was almost matching the lag-time each model had on their inputs over the output.

June 2017:

While comparing the months of June as they where unusual months every year, was observed that the June of 2017 in particular was the period of time with higher consumption over the four years. This month had also more variability than usual. Therefore, this month was selected for the second forecasting. During this period of time, there was the highest peak in consumption of the four years. During the first week of the month the consumption was rather common, considering it was summer, than significantly high. However, from the end of the second week on, the consumption raised. Therefore, the third week of the June 2017, three common days, and highes pick of consumption were the three periods of high consumption selected to perform the forecasting.

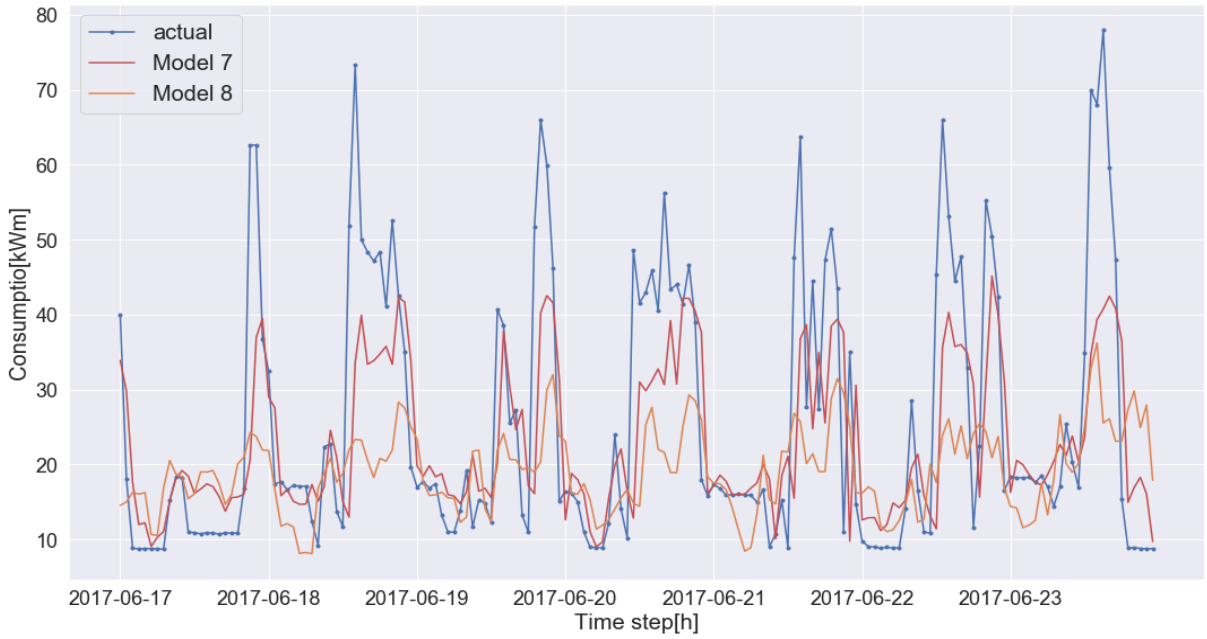


Figure 8.4: Third week of June 2017, forecasted by Model 7 and 8 and compared to the actual values.

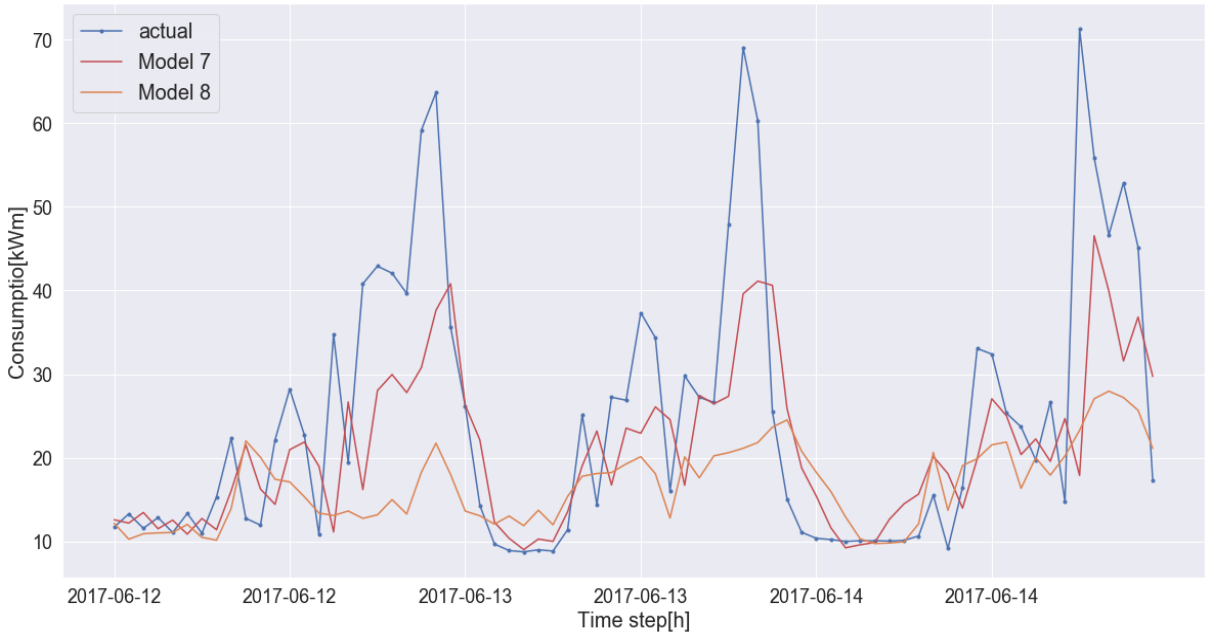


Figure 8.5: Days 12th, 13th and 14th of June 2017, forecasted by Model 7 and 8 and compared to the actual values.

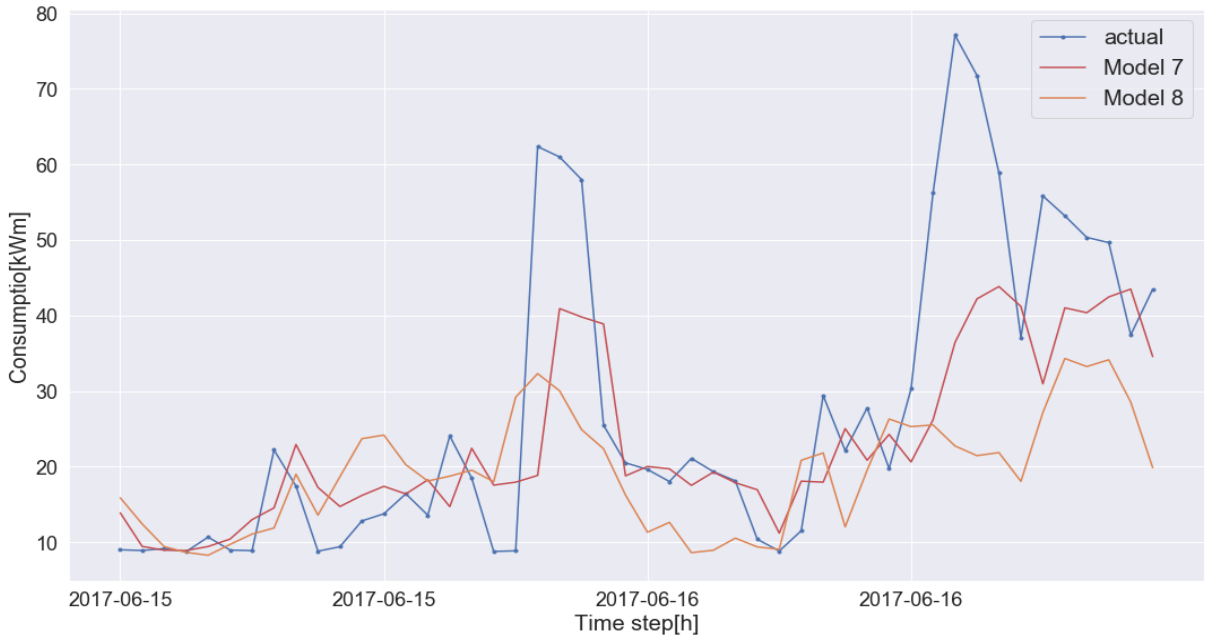


Figure 8.6: Peak on second week of June 2017, forecasted by Model 7 and 8 and compared to the actual values.

The results on forecasting specific periods of time of the month with most electrical consumption, showed how models acted with more accuracy at lower values than on peaks. In ***Fig.***8.4 is visible a similar curve but at three levels, the actual target values at the upper level and models 7 and 8 at a lower level being the orange curve the lowest. Therefore, it was confirmed that models were not able to reach the real values during high consume period. However, it seems as though they were able to forecast the rise and downs, variability which characterized this month too. At ***Fig.***8.6 is reflected how

the models were not able to achive such high values. Is interesting to point out how at the first image, is shown that both models practically every day reached forecasted values to which seemed impossible to arrive observing the results of May 2016. In major, the peaks during May of 2016 were found around 30-40kWm, values that on June 2017 were achieved by the models.

1st January, 2019:

The 1st of January of 2019 was the first day from which the information on the electrical consumption was unknown. Consequently, there were not real values to visualy analyse the accuracy of the models. Likewise, it was not possible to compare the performance of the models as the 8th was the only model able to forecast for a 24 hours period, without any information in consumption for the mentioned day.

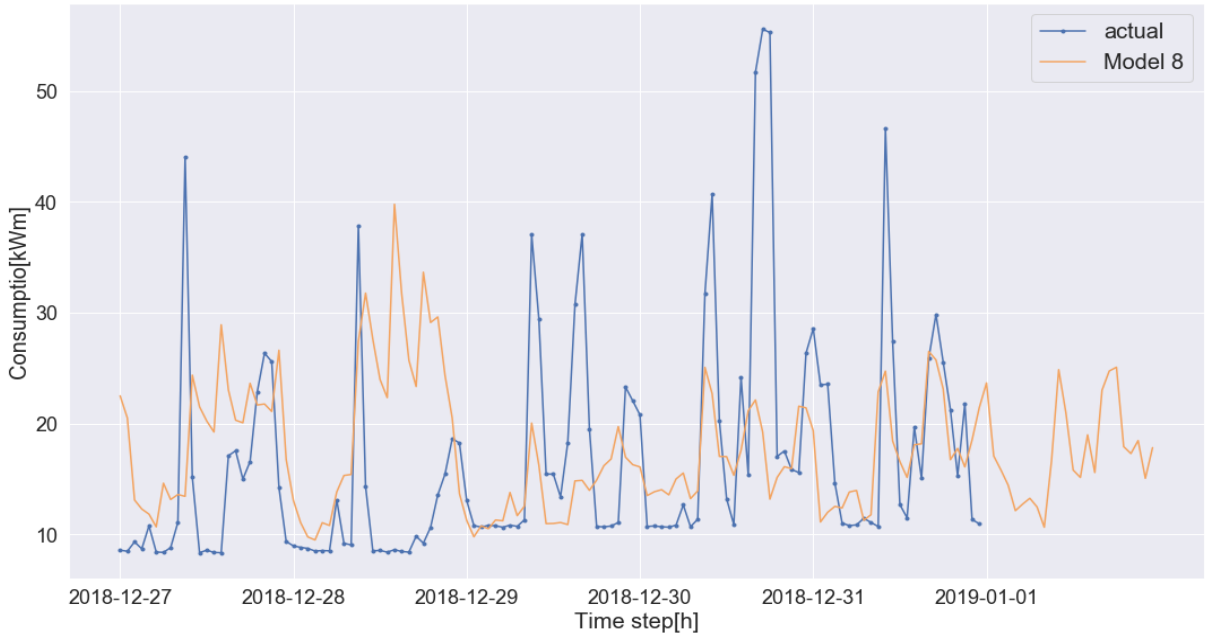


Figure 8.7: Last days of the data set followed by the following 24h, forecasted by Model 8.

The ***Fig***.8.7 show from the 27th to 31st of December 2018, the electric energy consumption the sensor registered and the forecast from Model 8. In addition, the forecast of Model 8 was added for the following 24 hours. The accuracy of values computed by the network for last days of December were not the best work the net was capable of, it showed better accuracy on previous experiments. This was caused by the high variability that the real consumption had. It was holiday time, consequently during some periods of time any member of the apartment was home as the continuous-low hours of consumption can be seen for instance in the midday time of 27th and 28th. In contrast there was a peak of consumption on midday time of 30th.

9 Cost Analysis

The total cost associated to the project was the combinations of the authors personal work and the equipment costs. All the work was done with open-source software. *Jupyter*, a web-based interactive computational environment to support data science and scientific computing across all programming languages was used as the *Python* environment. *Overleaf*, the on-line version of *Latex*, was used to write and design the dissertation.

9.1 Human labor cost

During 3 months the student had been working on the thesis; 6h/day for two months performing the research, analysis and computation needed, and an average of 8h/day for a month performing the writing and design of the dissertation. Plus some extra hours to finish the writing part and last experiments. Concluding in a total of 450h of work. Working as a professional added a price of 25€/h for the labor time.

Task	Time[h]	Cost
	294	6.650€
Dissertation	184	4.600€
Total	478	11.950€

Table 9.1: Personal cost.

9.2 Equipment costs

A P65 Creator MSI was the tool used for all the analysis, programming, computation and writing. The price of the device is 1.450€, with an estimated service life of 10 years computing a cost of 12,16€/month. Consequently, adding 36,48€ to the project cost.

9.3 Total cost

The total cost of the project has been; 11.986,48€.

10 Environmental impact

The performance of the project *per se*, had neither positive nor negative impact on the environment. However the implementation of an accurate forecasting of the electric energy demand at house-hold level can provide intelligence to smart meters (SM). SM systems are a part of micro-grids which encompass a variety of operational and energy measures including smart appliances, renewable energy resources and energy efficient resources [Net]. This technology allows smart grid vision and smart homes, which can optimize energy consumption and lower electricity bills. Consequently, it would cause a positive effect on the environment and at personal-level directly.

Conclusions and future

Conclusions

To sum up, through this thesis an ANN was build and trained for its subsequent forecast of electricity demand over a particular dwelling. First of all, an exploratory analysis of the data was performed, with the main purpose of finding the necessary insights on the data set in order to later build the network and its models. In addition, during the EDA an amount of missing values over the set, which needed to be removed for the correct functioning of the network, was found. Then, the building process was done in order to chose the ideal selection of hyperparameters for this specific forecasting problem. Probably the most work-loaded part of the project was the development test, where the first models were created. Moreover, the optimal hyperparameters from the previous selection were chosen. The first predictions were performed after the training of the network. Also after the training the last model of the project was created. Finally, an analysis on the forecasting was done by two models, subjected to distinct situations.

Initially, during the development of the models it was discovered that labels which appear to be obvious were already learnt by the network with no need of additional inputs, as identifying the season of the year for instance. Also, the wide range of possible values for each hyperparameter brought to light the importance of performing a robust hyperparameter searching.

Regarding the act of forecasting, it could be seen how a shallow network fed with consumption values sufficiently time-closed to the target values, and label inputs such as the week day, is able to reach an acceptable forecasting accuracy. However, this described model did not allow to achieve the main purpose of the project, to forecast the oncoming 24h of electric energy demand. During the creation of the last model, was learnt that there are several ways to analyse and quantify the accuracy of the network apart from its metrics and other direct error functions, visualizing the data points from different perspectives for instance. Finally, although the forecasting of the 24 hours period in advance performed the forecasting with certain amount of accuracy on the metric of evaluation, it showed weak response outside a central tendency of values. Regardless the level of values the central tendency had, whichever values the peaks were achieving, the forecasted points had difficulties on deviating from the central tendency and consequently were far to reach the peak levels.

Future work

The adversities and limits that have been appearing in the course of this project, opened the gates to possible improvements to apply on this thesis, as well as further projects. As improvements, in order to achieve a higher certainty on the forecasting;

- Perform a deep analyse on the forecasting of the peaks and lower level values. A starting point could be studying the influence of a wide range of activation function on the accuracy of the network.

- Study the daily trend and the analyse the optimal hyperparameters and inputs in order to achieve almost perfect accuracy during highly repeated periods of time such as sleeping-time.
- Work on a methodology which allows the possibility of including the immediate forecasted values into the inputs. This could allow to forecast 24h with the higher accuracy of model 7.
- Use of a DNN to perform same work and compare the difference in complexity and accuracy.

Regarding new projects, it would be interesting to perform the forecasting at appliance-level. Although the prediction of individual appliances consumption requires more time than forecasting the overall consumption, there can be a great save of energy. Depending on the type of appliance energy-saving action could be done on the studied devices. Moreover forecasting the demand of the most-consuming appliances provides an amount of high accurate demand forecasting, from devices with constant consumption. Then, the problem of the variability would remain in a lower level and amount of appliances. A related research would be the performance at household-level of the electrical demand forecasting with additional information such as meteorological data, to find more useful insight as temperature or rainfall correlations with the consumption of electricity.

Acknowledgments

I would like to express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement, and to the friends that have been by my side throughout my years of study and through the process of researching and writing this thesis.

Bibliography

[Mal17] Farhad Malik. "Understanding what really happens in a neural network; Neural Network Activation Function Types." In: (May, 2017). URL: https://medium.com/fintechexplained/neural-network-activation-function-types-a85963035196.

[Kha19] Rafay Khan. "Why Using Mean Squared Error(MSE) Cost Function for Binary Classification is a Bad Idea?." In: (Nov, 2019). URL: https://towardsdatascience.com/why-using-mean-squared-error-mse-cost-function-for-binary-classification-is-a-bad-idea-933089e90df7.

[Bro17] Jason Brownlee. "How to Configure the Number of Layers and Nodes in a Neural Network." In: (July, 2017). URL: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/.

[Lau17] Suki Lau. *Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning*. July, 2017. URL: https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1.

[Bro18] Jason Browlee. *Difference Between a Batch and an Epoch in a Neural Network*. July, 2018. URL: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/.

[DeF19] Robert R.F. DeFilippi. "Standardize or Normalize?." In: (April, 2019). URL: https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc.

[V17] Avinash Sharma V. "Understanding Activation Functions in Neural Networks." In: (march, 2017). URL: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

[Ked17] Chinmay D. Pai Kedar Potdar Taher S. Pardawala. "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers." In: *International Journal of Computer Applications* (October, 2017). URL: https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc.

[Koe18] Will Koehrsen. "Overfitting vs. Underfitting: A Conceptual Explanation." In: (January, 2018). URL: https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanation-d94ee20ca7f9.

[Bag18] Juan Ignacio Bagnato. "Breve Historia de las Redes Neuronales Artificiales." In: (September, 2018).

[Roj96] Raul Rojas. *Neural Networks: A Systematic Introduction*. Spring Science Business Media, 1996. ISBN: 3540605053.

[Xav10] Yoshua Bengio Xavier Glorot. "Understanding the difficulty of training deep feedforward neural networks." In: (2010).

[Guo14a] Michael Y. Hu Guoqiang Zhang B. Eddy Patuwo. "Forecasting with artificial neural networks: The state of the art." In: *International Journal of Forecasting* (July 2014), pp. 35–37.

[Guo14b] Michael Y. Hu Guoqiang Zhang B. Eddy Patuwo. "Forecasting with artificial neural networks: The state of the art." In: *International Journal of Forecasting* (July 2014), pp. 42–46.

[KB14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[MSK14] Kamal Malik, Harsh Sadawarti, and Gursharanjeet Kalra. "Comparative Analysis of Outlier Detection Techniques". In: *International Journal of Computer Applications* 97 (July 2014), pp. 12–21. DOI: 10.5120/17026-7318.

[Com16] European Comission. "Overview of European Electricity Markets". In: *European Comission* (2016).

[CM18] A. Gachagan C. E. Nwankpa W. Ijomah and S. Marchall. *Activation Functions: Comparison of trends in practice and research for deep learning*. 2018. arXiv: 1811.03378v1 [cs.LG].

[DeJ19] John DeJesus. "Point Biserial Correlation with Python." In: (2019). URL: https://towardsdatascience.com/point-biserial-correlation-with-python-f7cd591bd3b1.

[Goo19] Google. *Machine Learning Crash Course*. 2019. URL: https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[Nav19] Avinash Navlani. *Neural Network Models in R*. Dec. 2019. URL: https://www.datacamp.com/community/tutorials/neural-network-models-r.

[Raj19] Rahul Raj. *Java Deep Learning Cookbook*. Packt>, Nov. 2019. URL: https://subscription.packtpub.com/book/data/9781788995207/1/ch01lvl1sec02/deep-learningand-xa0-intuition.

[Eur] Eurostat. *Energy consumption and use by households*. URL: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20190620-1. (accessed: 03.01.2020).

[Net] Climate Technology Centre Network. *Smart grid*. URL: https://www.ctc-n.org/technologies/smart-grid.